目　次

第1章 典型前向神经网络

人工神经网络(Artificial Neural Network, 简称 ANN）是由大量简单的处理单元组成的非线性、自适应、自组织系统，它是在现代神经科学研究成果的基础上，试图通过模拟人类神经系统对信息进行加工、记忆和处理的方式，设计出的一种具有人脑风格的信息处理系统。人脑是迄今为止我们所知道的最完善最复杂的智能系统，它具有感知识别、学习、联想、记忆、推理等智能，人类智能的产生和发展经历了漫长的进化过程，而人类对智能处理的新方法的认识主要来自神经科学。虽然人类对自身脑神经系统的认识还非常有限，但已设计出像人工神经网路这样具有相当实用价值和较高智能水平的信息处理系统。

按其信息流向来分类，人工神经网络可以被分成前向网络和反馈网络。本章将对典型前向网络的结构、功能及其性能予以介绍，其中包括感知器、自适应线性元件和反向传播网络。

人工神经网络是通过计算机软件或电子线路硬件构成的。它是由最简单的人工神经元并联，或者再加上串联组成。人们通常用图形来表示网络系统的输入到输出的转化关系。单个神经元可以表示为如图 1.1 所示的模型结构，其中，神经元输入矢量用矩阵形式可以表示为：

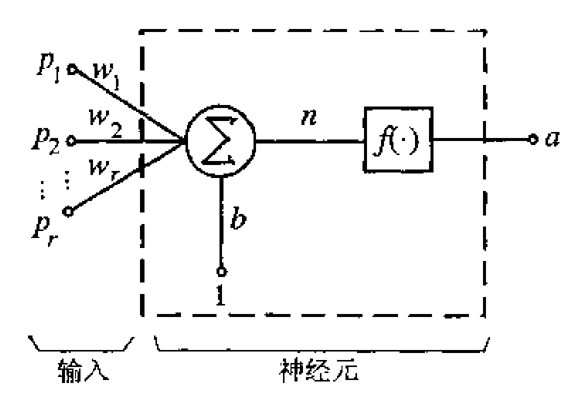


图 1.1 单个神经元模型结构

$$P = [p_1\ p_2 \cdots p_r]^T，\quad 权矩阵\ W = [w_1\ w_2 \cdots w_r]$$

单个神经元输出 A 与输入 P 之间的对应关系可表示为：

$$A = f\left(\sum_{j=1}^{r} w_j p_j + b\right) = f(W*P + b) \tag{1.1}$$

其中，b 称为阀值，或偏差，f 代表某种激活函数关系式。

可以说单个神经元就是一个多输入/单输出的系统。两个或更多的单神经元相并联则构成单层神经网络，如图 1.2 所示。

两个及其以上单层神经网络相级联则构成多层神经网络，如图 1.3 所示。

由于在网络输入/输出关系式中对所使用的变量采用了矩阵形式来表达，所以对于单层多输出神经网络，其表达形式与单个神经元时完全相同。如图 1.2 所表示的神经元个数为 s 的单层神经网络的输入/输出关系式可写为：

$$A_{s\times1} = F(W_{s\times r} * P_{r\times1} + B_{s\times1}) \tag{1.2}$$

一个多层神经网络的输入/输出关系，可以以单层神经网络的关系进行递推，将前一层的输出作为后一层的输入，并采用与单层的神经网络同样的书写方式即可方便地写出。

如图 1.3 所示的一个具有 3 层神经网络的输入/输出关系可写为：

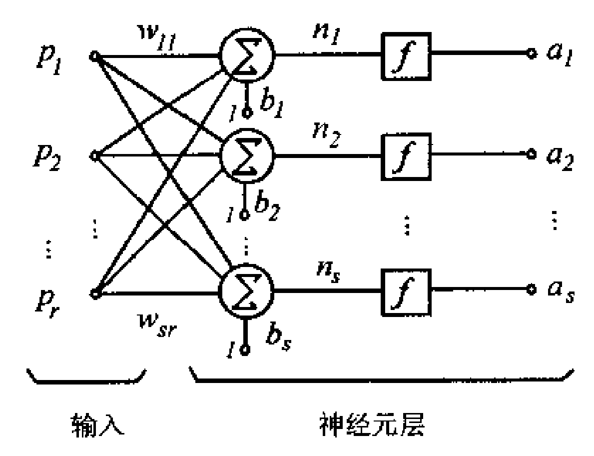
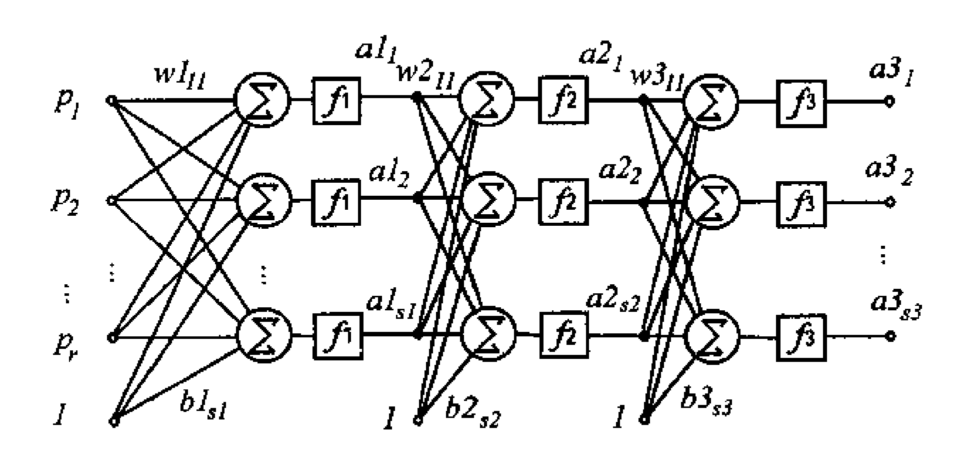
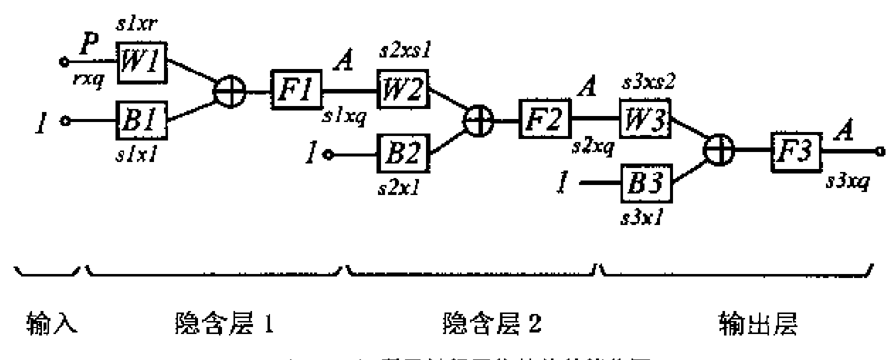


图 1.2 单层神经元网络模型结构图


(a) 三层的神经网络结构图



(b) (a) 所示神经网络结构的简化图

图 1.3 多层神经网络

$$
\begin{aligned}
A1 &= F1(W1*P+B1) \\
A2 &= F2(W2*A1+B2) \\
A3 &= F3(W3*A2+B3) \\
&= F3\{W3*F2[W2*F1(W1*P+B1)+B2]+B3\}
\end{aligned}
\tag{1.3}
$$

对于这种标准全联接的多层神经网络，更一般的简便作图是仅画出输入节点和一组隐

含层节点外加输出节点及其连线来示意表示，如图 1.4 所示。图中只标出输入、输出和权矢量，完全省去激活函数的符号。完整的网络结构则是通过具体的文字描述来实现的，如：网络具有一个隐含层，隐含层中具有 5 个神经元并采用 S 型激活函数，输出层采用线性函

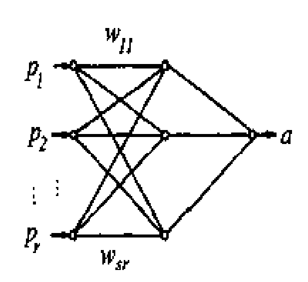


图 1.4 神经网络结构示意图

数，或者更简明的可采用“网络采用 2-5-1 结构”来描述，其中，2 表示输入节点数；5 表示隐含层节点数；1 为输出节点数。如果不对激活函数作进一步的说明，则意味着隐含层采用 S 型函数，而输出层采用线性函数。

特别值得强调的是，在设计多层网络时，隐含层的激活网络应采用非线性的，否则多层网络的计算能力并不比单层网络更强。因为在采用线性激活函数的情况下，由于线性激活函数的输出就等于网络的加权输入和，即 $A=W*P+B$，如果将偏差作为一组权值归为 W 中统一处理，两层网络的输出 $A2$ 则可以写为：

$$\begin{aligned} A2 &= F2(W2 * A1) \\ &= W2 * F1(W1 * P) \\ &= W2 * W1 * P \\ &= W * P \end{aligned} \tag{1.4}$$

其中，$F1 = F2 = F$ 为线性激活函数。上式表明两层线性网络的输出计算等效于具有权矢量为 $W = W2*W1$ 的单层线性网络的输出。

由此可见，人工神经网络工作时，所表现出的就是一种计算，利用人工神经网络求解问题时所利用的也正是网络输入到网络输出的某种关系式。与其他求输入/输出关系式方法不同的是，神经网络的输入/输出关系式是根据网络结构写出的，并且网络权值的设计往往是通过训练而不是根据某种性能指标计算出来的。所以，应用神经网络解决实际问题的关键在于设计网络，而网络的设计主要包括两方面的内容，一是网络结构，另一个是网络权值的确定。第一个方面涉及到对不同网络结构所具有的功能及其本质的认识，第二个方面涉及到对不同网络权值训练所用的学习规则的掌握。所以无论是作为学习、应用，还是作为更深层次的理论研究，这两步都是最重要的。正因为如此，人们对人工神经网络进行分类时，也是有两种方法，一种是根据网络结构分为前向网络和反馈网络两大类，另一种是按训练权值方法分为监督式（或有教师）网络和无监督式（无教师）网络两种。

在神经网络的输入与输出之间所存在的关系函数，类似于控制系统中的转移函数，称为激活函数，由它决定网络的线性和非线性，它是网络特性及功能的关键所在。在控制系统中，一般对控制器的设计是根据某个性能指标来设计控制器的结构或参数，与此不同，在神经网络信息处理系统中，设计者是根据期望网络完成的目标（或任务）以及某种调整网络权值所采用的学习规则来确定网络的结构与权值。由于网络的权值一般不是计算出而是训练出的，所以利用神经网络解决问题时，对于采用监督式学习的网络，网络的设计要

经过两个阶段，首先是网络权值的训练（即学习并修正）阶段，然后才是网络的运行工作阶段。而对于采用非监督式学习的网络，则将网络权值的训练与工作合二为一，即在网络工作的同时，调整网络的权值，所以神经网络具有自学习、自适应的特性。

一般而言，由人工神经网络组成的信息处理系统所具有的特性为：

1）非线性

神经网络对非线性控制领域问题的解决带来了希望，这主要是由神经网络理论上能够任意逼近非线性连续有理函数的能力所决定的。神经网络还能够比其他逼近方法得到更加易得的模型。

2）并行分布处理

神经网络具有一个使其自身进行并行实施的高度并行结构，如此的实现结构使其能够达到比常规方法所获得结果具有更高的容错性。另外，虽然一个神经网络中的基本处理单元是非常简单的结构，然而将其进行并行实现的连接，则产生了极快的整体处理效果。

3）硬件实现

神经网络不仅能够进行并行处理，还可以通过引入大规模集成电路将其进行硬件实现，这将带来附加的速度，并且可以增加应用网络的规模。

4）学习与自适应

神经网络是通过采用被研究系统的数据记录进行训练而获得的。对于一个训练好的网络，对其输入训练中未知的数据时，具有很好的泛化能力。

5）数据融合

神经网络能够同时操作定量和定性的数据，这一特性使神经网络可以处理处在传统的系统工程（定量数据）与人工智能领域的处理技术（符号数据）之间的问题。

6）多变量系统

神经网络本身就是一个多输入/多输出系统，这个系统为解决复杂的多变量系统的建模和控制等问题开辟了一条新的途径。

1.1 感知器网络

我们已经知道，人工神经网络是在人类对其大脑神经网络认识理解的基础上人工构造的能够实现某种功能的神经网络。它是理论化的人脑神经网络的数学模型，是基于模仿大脑神经网络结构和功能而建立的一种信息处理系统。人工神经网络吸取了生物神经网络的许多优点，它具有高度的并行性、非线性的全局作用，以及良好的容错性与联想记忆功能，并且具有很强的自适应、自学习能力。随着人工神经网络技术的不断发展，其应用领域也在不断拓展，主要应用于模式信息处理、函数逼近和模式识别，以及联想记忆、最优化问题计算和自适应控制等方面。在前向网络中，最典型的网络是反向传播网络，不过，它也是在最早的人工神经网络——感知器基础上发展起来的，并且在线性分类问题中，感知器仍然发挥着重要的作用。

1.1.1 感知器的网络结构及其功能

最早用数学模型对神经系统中的神经元进行理论建模的是美国心理学家麦卡洛克(W. McCulloch)和数学家皮茨（W. Pitts）。他们于 1943 年在分析和研究了人脑细胞神经元后用电路构成了简单的神经网络数学模型（简称 MP 模型）。感知器（Perceptron）是由美国计算机科学家罗森布拉特（F. Rosenblatt）于 1957 年提出的。感知器是在 MP 模型的基础上，加上学习功能，使其权值可以连续调节的产物。它是一个具有一层神经元、采用阈值激活函数的前向网络。

感知器的网络结构是由单层 s 个感知神经元，通过一组权值 $\{w_{ij}\}$ ($i = 1, 2, \ldots, s$； $j = 1, 2, \ldots, r$)与 r 个输入相连组成。对于具有输入矢量 $P_{r\times q}$ 和目标矢量 $T_{s\times q}$ 的感知器网络的简化结构如图 1.5 所示。阈值激活函数如图 1.6 所示。

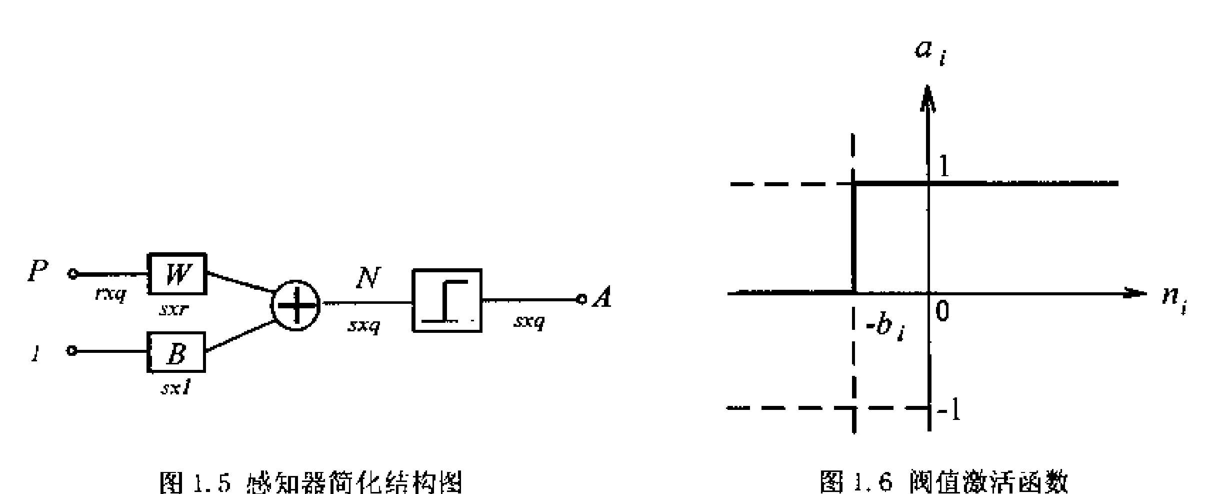


图 1.5 感知器简化结构图　　　　图 1.6 阈值激活函数

感知器输入/输出函数关系式为：

$$A = \begin{cases} 1 & \sum W * P + b \geq 0 \\ 0 & \sum W * P + b < 0 \end{cases} \tag{1.5}$$

由上式可知，通过对网络权值的训练，可以使感知器对一组输入矢量的响应达到元素为 0 或 1 的目标输出，从而实现对输入矢量分类的目的。

感知器的这一功能可以通过在输入矢量空间里的作图来加以解释。以输入矢量 $r = 2$ 为例，对于选定的权值 w_1、w_2 和 b，可以在以输入矢量 p_1 和 p_2 分别作为横、纵坐标的输入平面内画出 $W*P + b = 0$，即 $w_1p_1+w_2p_2+ b = 0$ 的轨迹，它是一条直线，此直线上的以及线以上部分的所有 p_1、p_2 值均使 $w_1p_1+w_2p_2+b \geq 0$，这些点若通过由 w_1、w_2 和 b 构成的感知器则使其输出为 1；该直线以下部分的点则使感知器的输出为 0，如图 1.7 所示。

所以当采用感知器对不同的输入矢量进行期望输出为 0 或 1 的分类时，其问题则转化为：对于已知输入矢量所处输入平面的不同点的位置，设计感知器的权值 W 和 b，将由 $W*P + b = 0$ 的直线放置在适当的位置上使输入矢量按期望输出值进行上下分类。推而广之，阈值函数通过将输入矢量的 r 维空间分成若干区域而使感知器具有将输入矢量分类的能力。对于不同的输入神经元 r 和输出神经元 s 组成的感知器，当采用输入矢量空间的作

图法来解释网络功能时，其分类的一般情况可以总结为：

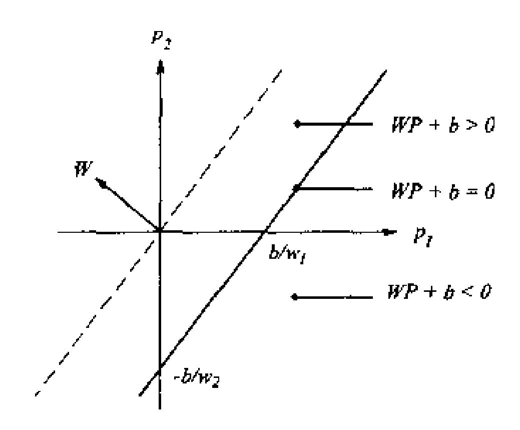


图 1.7 感知器的图形解释

①当网络输入为单个节点，输出也为单个神经元，即 $r = 1$，$s = 1$ 时，感知器是以点作为输入矢量轴线上的分割点；

②当网络输入为两个节点，即 $r = 2$ 时，感知器是以线对输入矢量平面进行分类；其中：当 $s = 1$，分类线为一条；$s = 2$，分类线为两条，依此类推。输出神经元个数 s 决定分类的直线数，可分成的种类数为 2^s；

③当网络输入为三个节点，即 $r = 3$ 时，感知器是以平面来分割输入矢量空间，而且用来进行空间分割的平面个数等于输出神经元个数 s。

1.1.2 感知器权值的学习规则与训练

学习规则是用来计算新的权值矩阵 W 及新的偏差 B 的算法。感知器利用其学习规则来调整网络的权值，以便使网络对输入矢量的响应达到数值为 0 或 1 的目标输出。

对于输入矢量 P、输出矢量 A、目标矢量为 T 的感知器网络，感知器的学习规则是根据以下输出矢量可能出现的三种情况来进行参数调整的。

①如果第 i 个神经元的输出是正确的，即有 $a_i = t_i$，那么与第 i 个神经元联接的权值 w_{ij} 和偏差 b_i 保持不变；

②如果第 i 个神经元的输出是 0，但期望输出为 1，即有 $a_i = 0$，而 $t_i = 1$，此时权值修正算法为：新的权值 w_{ij} 为旧的权值 w_{ij} 加上输入矢量 p_j；类似地，新的偏差 b_i 为旧偏差 b_i 加上它的输入 1；

③如果第 i 个神经元的输出为 1，但期望输出为 0，即有 $a_i = 0$，而 $t_i = 1$，此时权值修正算法为：新的权值 w_i 等于旧的权值 w_i 减去输入矢量 p_j；类似地，新的偏差 b_i 为旧偏差 b_i 减去 1。

由上面分析可以看出，感知器学习规则的实质为：权值的变化量等于正负输入矢量。具体算法总结如下：

对于所有的 i 和 j, $i = 1, 2, \cdots, s$；$j = 1, 2, \cdots, r$，感知器修正权值公式为：

$$\begin{aligned}\Delta w_{ij} &= (t_i - a_i) \times p_j \\ \Delta b_i &= (t_i - a_i) \times 1\end{aligned} \quad (1.6)$$

用矢量矩阵来表示为：

$$
\begin{aligned}
W &= W + EP^{\mathrm{T}} \\
B &= B + E
\end{aligned}
\tag{1.7}
$$

此处，E 为误差矢量，有 $E = T - A$ 。

感知器的学习规则属于梯度下降法。已被证明：如果解存在，则算法在有限次的循环迭代后可以收敛到正确的目标矢量。

要使前向神经网络模型实现某种功能，必须对它进行训练，让它逐步学会要做的事情，并把所学到的知识记忆在网络的权值中。人工神经网络权值的确定不是通过计算，而是通过网络的自身训练来完成的。这也是人工神经网络在解决问题的方式上与其他方法的最大不同点。借助于计算机的帮助，几百次甚至上千次的网络权值的训练与调整过程能够在很短的时间内完成。

感知器的训练过程如下：

在输入矢量 P 的作用下，计算网络的实际输出 A，并与相应的目标矢量 T 进行比较，检查 A 是否等于 T，然后用比较后的误差 E，根据学习规则进行权值和偏差的调整；重新计算网络在新权值作用下的输入，重复权值调整过程，直到网络的输出 A 等于目标矢量 T 或训练次数达到事先设置的最大值时训练结束。

若网络训练成功，那么训练后的网络在网络权值的作用下，对于被训练的每一组输入矢量都能够产生一组对应的期望输出；若在设置的最大训练次数内，网络未能够完成在给定的输入矢量 P 的作用下，使 $A = T$ 的目标，则可以通过改用新的初始权值与偏差，并采用更长训练次数进行训练，或分析一下所要解决的问题是否属于那种由于感知器本身的限制而无法解决的一类。

感知器设计训练的步骤可总结如下：

①对于所要解决的问题，确定输入矢量 P，目标矢量 T，并由此确定各矢量的维数以及确定网络结构大小的神经元数目：r, s 和 q；

②参数初始化：a) 赋给权矢量 W 在(-1, 1)的随机非零初始值；

b) 给出最大训练循环次数；

③网络表达式：根据输入矢量 P 以及最新权矢量 W，计算网络输出矢量 A；

④检查：检查输出矢量 A 与目标矢量 T 是否相同，如果是，或已达最大循环次数，训练结束，否则转入⑤；

⑤学习：根据(1.7)式感知器的学习规则调整权矢量，并返回③。

下面给出例题来进一步了解感知器解决问题的方式，掌握设计训练感知器的过程。

例 1.1 考虑一个简单的分类问题。

设计一个感知器，将二维的四组输入矢量分成两类。

输入矢量为：P = [-0.5 -0.5 0.3 0;

-0.5 0.5 -0.5 1];

目标矢量为：T = [1.0 1.0 0 0],

解：通过前面对感知器图解的分析可知，感知器对输入矢量的分类实质是在输入矢量空间用 $W*P + B = 0$ 的线性表达式对其进行分割而达到分类的目的。根据这个原理，对此例中二维四组输入矢量的分类问题，可以用下述不等式组来等价表示出：

$$\begin{cases}-0.5w_1-0.5w_2+b\geq 0 & (使t_1=1成立)\\ -0.5w_1+0.5w_2+b\geq 0 & (使t_2=1成立)\\ 0.3w_1-0.5w_2+b<0 & (使t_3=0成立)\\ w_2+b<0 & (使t_4=0成立)\end{cases}$$

实际上可以用代数求解法来求出上面不等式中的参数 w_1、w_2 和 b。经过迭代和约简，可得到解的范围为：

$$\begin{cases}w_1<0\\ 0.8w_1<w_2<-w_1\\ w_1/3<b<-w_1\\ b<-w_2\end{cases}$$

一组可能解为：

$$\begin{cases}w_1=-1\\ w_2=0\\ b=-0.1\end{cases}$$

而当采用感知器神经网络来对此题进行求解时，意味着采用具有阈值激活函数的神经网络，按照问题的要求设计网络的模型结构，通过训练网络权值 $W=[w_{11}\ w_{12}]$ 和 b，并根据学习算法和训练过程进行程序编程，然后运行程序，让网络自行训练其权矢量，直至达到不等式组的要求。

鉴于输入和输出目标矢量已由问题本身确定，所以所需实现其分类功能的感知器网络结构的输入节点 r，以及输出节点数 s 已被问题所确定而不能任意设置。

根据题意，网络结构图如图 1.8 所示。

由此可见，对于单层网络，网络的输入神经元数 r 和输出神经元数 s 分别由输入矢量 P 和目标矢量 T 唯一确定。网络的权矩阵的维数为：$W_{s\times r}$，$B_{s\times 1}$，权值总数为 $s\times r$ 个，偏差个数为 s 个。

在确定了网络结构，设置了最大循环次数并赋予权值初始值后，设计者便可方便地利用适当软件，根据题意以及感知器的学习、训练过程来编写程序，并通过计算机对权值的反复训练与调整，最终求得网络的结构参数。

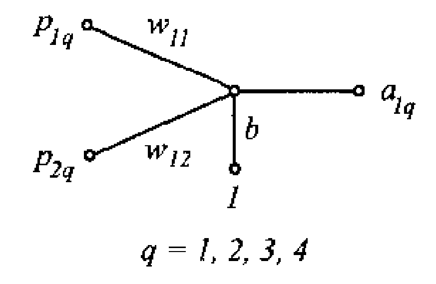


图 1.8 网络结构图

上面的结果似乎表明只要增加输出神经元数 s，就能够解决任意数目的分类问题。事实上，对于输出为 0 或 1 的感知器，其功能可以典型化为对各种逻辑运算的实现。当网络

具有 r 个二进制输入分量时，最大不重复的输入矢量有 2^r 组， 其输出矢量所能代表的逻辑功能总数为 2^{2^r} 。不幸的是，感知器不能够对任意的输入/输出对应的关系（即任意逻辑运算）进行实现，它只能够实现那些在图 1.7 中用直线、平面等进行线性分类的问题，即感知器不能够对线性不可分的输入矢量进行分类。所谓线性可分，是指输入及输出点集是几何可分的。对于两个输入的情形而言，输入及输出点集可用直线来分割：对于三个输入而言，可用平面来分割：依此类推，对于 r 个输入情形，线性可分是指可用 $r-1$ 维超平面来分割此 r 维空间中的点集。

最具有代表性的线性不可分的问题就是“异或”问题。逻辑运算中的“异或”功能，就是当输入两个二进制数同时为 0 或同时为 1 时，其输出为 0，只有当两个输入中有一个为 1，而另一个为 0 时，结果为 1 。若希望用感知器来实现此功能的操作，其输入应为两个二进制数所有可能情况的组合（共 4 种），然后根据“异或”逻辑功能，对应出各输入 P 下的目标输出 T：

$$P=\begin{bmatrix}0 & 0 & 1 & 1\\ 0 & 1 & 0 & 1\end{bmatrix},\quad T=[0110]$$

所要求解的是：设计一个单层感知器对输入矢量按期望的输出矢量进行分类。

根据图解，“异或”问题则是要求用一条直线将平面上的四个点分成两类。输入矢量平面上四点的位置如图 1.9 所示，其中，“×”表示希望将其输入分为 1 类：“o”表示对应的目标输出为 0。很显然，对于这样的四点位置，想用一条直线把相同类的期望输入区分开来是不可能的。实际上，从逻辑运算入手，可写出感知器对所有的四组二元素输入所对应的全部 16 种输出的情况，它们分别代表“与”、“或”、“非”、“与非”、“或非”、“异或”、“异或非”等逻辑运算功能。在 16 种逻辑操作功能中，只有“异或($T=[\,0\ 1\ 1\ 0\,]$)”和“异或非($T=[\,1\ 0\ 0\ 1\,]$)”是线性不可分的，其他 14 中逻辑功能均为线性可分，它们都可以用单层感知器来实现。

在实际的逻辑运算中，随着运算变量数目的增加（即输入矢量 r 的增加），线性可分的情况所占比例是相当少的。表 1.1 给出了不同的输入 r 时，线性可分的逻辑运算个数与逻辑运算总数的情况，从中可以看出，采用感知器只能解决很少部分的分类问题，它的应用存在着相当的局限性。

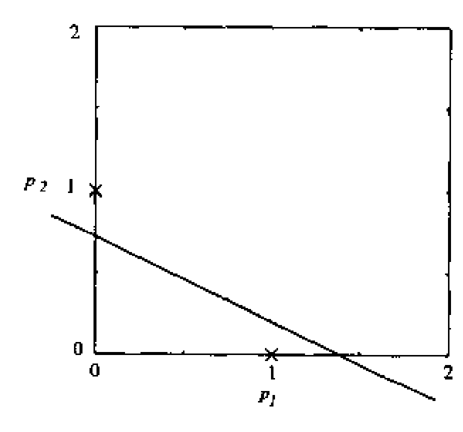


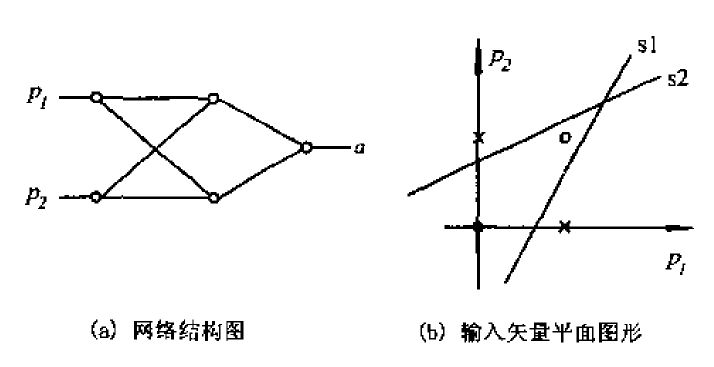
图 1.9 “异或”问题的图形表示

表 1.1 不同输入 r 时线性可分的功能数

r	逻辑运算总数 2^{2^r}	线性可分功能数
1	4	4
2	16	14
3	256	104
4	65536	1882
5	4.3×10^9	94572
6	1.8×10^{19}	5028134

感知器的线性可分性限制是个严重的问题。60 年代末，人们曾致力于该问题的研究，并找到了解决问题的办法，即变单层网络结构为多层网络结构。这实际上是把感知器的概念拓展化了。对于这样的 “异或” 问题，我们可以用两层网络结构，并在隐含层中采用两个神经元，即用两条判决直线 $s1$ 和 $s2$ 来解决。如图 1.10(a)所示。使 $s1$ 线下部分为 1，线上部分为 0，而 $s2$ 线的上部为 1，下部为 0，而在输出层中使图中阴影部分为 0 或 1，即可使“异或” 功能得以实现，而且其实现的可能有许多种。研究表明，两层的阀值网络可以实现任意的二值逻辑函数，且输入值不仅限于二进制数，可以是连续数值。需要指出的是，训练阀值型激活函数组成的两层感知器，所采用的修正权值的学习规则，仍然是单层感知器的学习规则，只是对两个单层网络分别各训练一次而已，隐含层的目标矢量可以从 14 个线性可分的矢量中选择那些将坐标平面上的 4 个点分别进行 1 个点和 3 个点的分类，并且，两条线的单点分类分别取处于对角线上点的目标矢量，例如可选择 $s1 = [\ 0\ 1\ 0\ 0\ ]$，$s2 = [\ 0\ 0\ 1\ 0\ ]$，它们的分类组成图 1.10（b）所示的分类结果。这样可以避免由隐含层的目标矢量又形成“异或” 问题，即隐含层的目标矢量的选取并不是任意的。实际上，通过分析，可以得知，$s1$ 和 $s2$ 各自不同的选择方案可组成的隐含层的目标矢量的总数只有 16 种。

另外，解决“异或” 问题的方法当然是采用具有连续可微激活函数作为隐含层函数的反向传播网络。与感知器的不同在于，那里对权值训练所采用的学习规则是误差的反向传播算法。



(a) 网络结构图　　(b) 输入矢量平面图形

图 1.10 “异或” 问题的一种解决方案

1.2 自适应线性元件

自适应线性元件（Adaptivc Linear Element，简称 Adaline）也是早期神经网络模型之一，它是由美国的威德罗（B. Widrow）和霍夫（M. Hoff）于 1959 年首先提出的。它与感知器的主要不同之处在于其神经元采用的是线性激活函数，这允许输出可以是任意值，而不仅仅只是像感知器中那样只能取 0 或 1。另外，它采用的是 W-H 学习法则，也称最小均方差(LMS)规则对权值进行训练，从而能够得到比感知器更快的收敛速度和更高的精度。

自适应线性元件的主要用途是线性逼近一个函数而进行模式联想。另外，它还适用于信号处理滤波、预测、模型识别和控制。

1.2.1 自适应线性神经元模型和结构

一个具有 r 个输入的自适应线性神经元模型如图 1.11 所示。这个神经元有一个线性激活函数，被称为 Adaline，如图 1.11(a)所示。和感知器一样，偏差可以用来作为网络的另一个可调参数，提供额外可调的自由变量以获得期望的网络特性。线性神经元可以训练网络学习一个与之对应的输入/输出的函数关系，或线性逼近任意一个非线性函数，但它不能产生任何非线性的计算特性。

当自适应线性网络由 s 个神经元相并联形成一层网络，此自适应线性神经网络又称为 Madaline，如图 1.11(b)所示。

W-H 规则仅能够训练单层网络，但这并不是什么严重问题。如前面所述，单层线性网络与多层线性网络具有同样的能力，即对于每一个多层线性网络，都具有一个等效的单层线性网络与之对应。

线性激活函数使网络的输出等于加权输入和加上偏差，如图 1.12 所示。此函数的输入/输出关系为：

$$A = f(W*P+b) = W*P + b \tag{1.8}$$

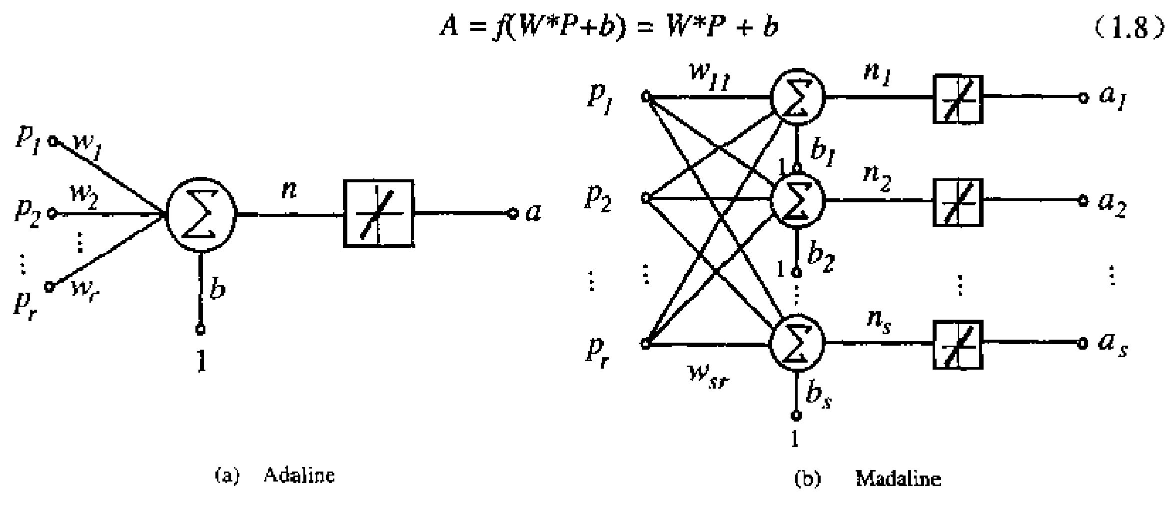


图 1.11 自适应线性神经网络的结构

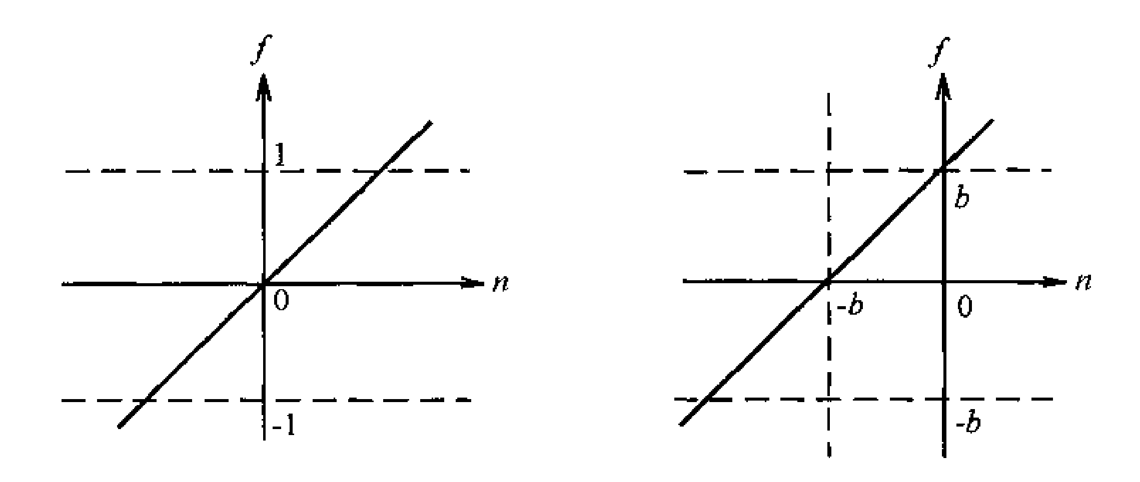


图 1.12 线性激活函数

1.2.2 W-H 学习规则及其网络的训练

W-H 学习规则是由威德罗和霍夫提出的用来修正权矢量的学习规则，所以用他们两人姓氏的第一个字母来命名。W-H 可以用来训练一层网络的权值和偏差使之线性地逼近一个函数而进行模式联想(Pattern Association)。

定义一个线性网络的输出误差函数为：

$$E(W,B)=\frac{1}{2}[T-A]^2=\frac{1}{2}[T-WP-B]^2 \tag{1.9}$$

由(1.9)式可以看出：线性网络具有抛物线型误差函数所形成的误差表面，所以只有一个误差最小值。通过 W-H 学习规则来计算权值和偏差的变化，并使网络误差的平方和最小化，总能够训练一个网络的误差趋于这个最小值。另外，很显然，$E(W,B)$只取决于网络的权值及目标矢量。我们的目的是通过调节权矢量，使 $E(W,B)$达到最小值。所以在给定 $E(W,B)$后，利用 W-H 学习规则修正权矢量和偏差矢量，使 $E(W,B)$从误差空间的某一点开始，沿着 $E(W,B)$的斜面向下滑行。根据梯度下降法，权矢量的修正值正比于当前位置上 $E(W,B)$的梯度，对于第 i 个输出节点有：

$$\Delta w_{ij}=-\eta\frac{\partial E}{\partial w_{ij}}=\eta[t_i-a_i]p_j \tag{1.10}$$

或表示为：

$$\Delta w_{ij}=\eta\delta_i p_j \tag{1.11a}$$

$$\Delta b_i=\eta\delta_i \tag{1.11b}$$

这里δ_i定义为第 i 个输出节点的误差：

$$\delta_i = t_i - a_i \tag{1.12}$$

(1.11)式称为 W-H 学习规则，又叫 δ 规则，或为最小均方差算法(LMS)。W-H 学习规则的权值变化量正比于网络的输出误差及网络的输入矢量。它不需求导数，所以算法简单，又具有收敛速度快和精度高的优点。

(1.11)式中 η 为学习速率。在一般的实际运用中，η 通常取一接近 1 的数，或取值为：

$$\eta = 0.99 * \frac{1}{\max[\det(P * P^{T})]} \tag{1.13}$$

这样的选择可以达到既快速又正确的结果。

自适应线性元件的网络训练过程可以归纳为以下三个步骤：

①表达：计算训练的输出矢量 $A = W*P + B$，以及与期望输出之间的误差 $E = T - A$；

②检查：将网络输出误差的平方和与期望误差相比较，如果其值小于期望误差，或训练已达到事先设定的最大训练次数，则停止训练。否则继续；

③学习：采用 W-H 规则计算新的权值和偏差，并返回到①。

每进行一次上述三个步骤，被认为是完成一个训练循环次数。

如果网络训练获得成功，那么当一个不在训练中的输入矢量输入到网络中时，网络趋于产生一个与其相联想的输出矢量。这个特性被称为泛化，这在函数逼近以及输入矢量分类的应用中是相当有用的。如果经过训练，网络仍不能达到期望目标，可以有两种选择：或检查一下所要解决的问题，是否适用于线性网络；或对网络进行进一步的训练。

虽然只适用于线性网络，W-H 规则仍然是重要的，因为它展现了梯度下降法是如何来训练一个网络的，此概念后来发展成反向传播法，使之可以训练多层非线性网络。

1.3 反向传播网络

1.3.1 反向传播网络模型与结构

反向传播网络（Back-Propagation Network，简称 BP 网络）是对非线性可微分函数进行权值训练的多层前向网络。在人工神经网络的实际应用中，80%~90%的人工神经网络模型是采用 BP 网络或它的变化形式，它主要用于以下几个方面：①函数逼近：用输入矢量和相应的输出矢量训练一个网络逼近一个函数；②模式识别：用一个特定的输出矢量将它与输入矢量联系起来；③分类：把输入矢量以所定义的合适方式进行分类；④数据压缩：减少输出矢量维数以便于传输或存储。可以说，BP 网络是人工神经网络中前向网络的核心内容，体现了人工神经网络最精华的部分。在人们掌握反向传播网络的设计之前，感知器和自适应线性元件都只能适用于对单层网络模型的训练，只是在 BP 网络出现后才得到了进一步拓展。

一个具有 r 个输入和一个隐含层的神经网络模型结构如图 1.13 所示。

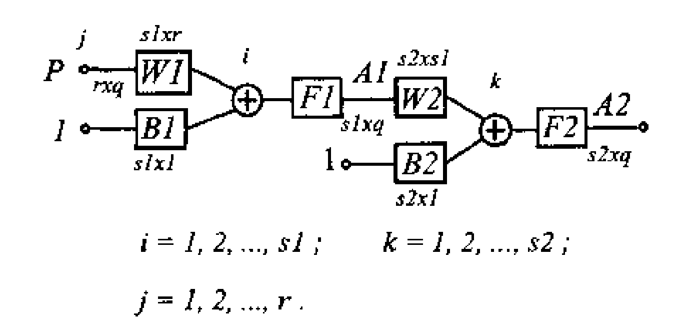


图 1.13 具有一个隐含层的神经网络模型结构图

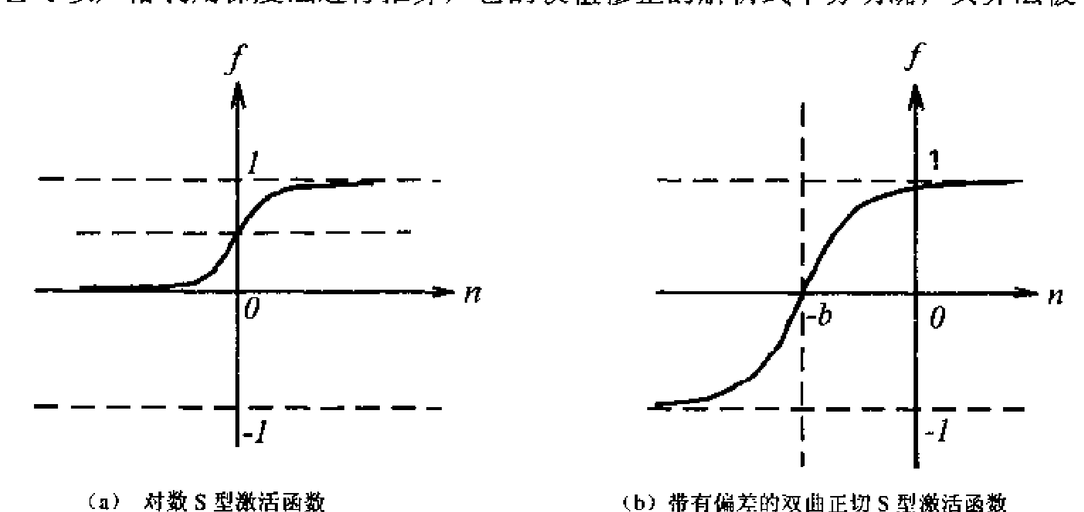
感知器和自适应线性元件的主要差别在激活函数上：前者是二值型的，后者是线性的。反向传播网络具有一层或多层隐含层，除了在多层网络结构上与前面已介绍过的模型有不同外，其主要差别还表现在激活函数上。反向传播网络的激活函数必须是处处可微的，所以它就不能采用二值型的阀值函数{0, 1}或符号函数{-1, 1}，反向传播网络经常使用的是 S 型激活函数。此种激活函数常用对数或双曲正切等一类 S 形状的曲线来表示，如对数 S 型激活函数关系为：

$$f=\frac{1}{1+\exp[-(n+b)]} \tag{1.14}$$

而双曲正切 S 型曲线的输入/输出函数关系为：

$$f=\frac{1-\exp[-2(n+b)]}{1+\exp[-2(n+b)]} \tag{1.15}$$

图 1.14 所示的是对数 S 型激活函数的图形。可以看到 $f(\cdot)$ 是一个连续可微的函数，它的一阶导数存在。对于多层网络，这种激活函数所划分的区域不再是线性划分，而是由一个非线性的超平面组成的区域。它是比较柔和、光滑的任意界面，因而它的分类比线性划分精确、合理，这种网络的容错性较好。另外一个重要的特点是由于激活函数是连续可微的，它可以严格利用梯度法进行推算，它的权值修正的解析式十分明确，其算法被称为



(a) 对数 S 型激活函数　　(b) 带有偏差的双曲正切 S 型激活函数

图 1.14 BP 网络 S 型激活函数

误差反向传播法，也简称 BP 算法，这种网络也称为 BP 网络。

因为 S 型函数具有非线性放大系数功能，它可以把输入从负无穷大到正无穷大的信号，变换成-1 到 1 之间输出，对较大的输入信号，放大系数较小；而对较小的输入信号，放大系数则较大，所以采用 S 型激活函数可以去处理和逼近非线性的输入/输出关系。

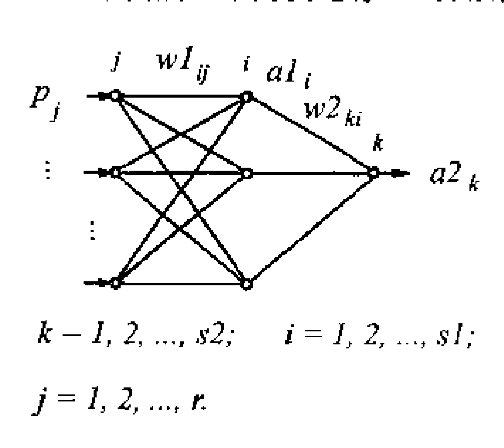
不过，如果在输出层采用 S 型函数，输出则被限制到一个很小的范围了，若采用线性激活函数，则可使网络输出任何值。所以只有当希望对网络的输出进行限制，如限制在 0 和 1 之间，那么在输出层应当包含 S 型激活函数，在一般情况下，均是在隐含层采用 S 型激活函数，而输出层采用线性激活函数。

1.3.2 BP 算法

BP 网络的产生归功于 BP 算法的获得。BP 算法属于 δ 算法，是一种监督式的学习算法。其主要思想为：对于 q 个输入学习样本：P^1，P^2，…，P^q，已知与其对应的输出样本为：T^1，T^2，… T^q。学习的目的是用网络的实际输出 A^1，A^2，…，A^q 与目标矢量 T^1，T^2，… T^q 之间的误差来修改其权值，使 A^n $(n = 1, 2, ..., q)$与期望的 T^n 尽可能的接近，即：使网络输出层的误差平方和达到最小。它是通过连续不断地在相对于误差函数斜率下降的方向上计算网络权值和偏差的变化而逐渐逼近目标的。每一次权值和偏差的变化都与网络误差的影响成正比，并以反向传播的方式传递到每一层的。

BP 算法是由两部分组成：信息的正向传递与误差的反向传播。在正向传播过程中，输入信息从输入经隐含层逐层计算传向输出层，每一层神经元的输出作用于下一层神经元的输入。如果在输出层没有得到期望的输出，则计算输出层的误差变化值，然后转向反向传播，通过网络将误差信号沿原来的连接通路反传回来修改各层神经元的权值直至达到期望目标。

为了明确起见，现以图 1.15 所示两层网络为例进行 BP 算法推导。



$k = 1, 2, ..., s2$;　　$i = 1, 2, ..., s1$;

$j = 1, 2, ..., r$.

图 1.15　具有一个隐含层的简化网络图

设输入为 P，输入神经元有 r 个，隐含层内有 $s1$ 个神经元，激活函数为 $F1$，输出层内有 $s2$ 个神经元，对应的激活函数为 $F2$，输出为 A，目标矢量为 T 。

1）信息的正向传递

(1) 隐含层中第 i 个神经元的输出为

$$a1_i = f1(\sum_{j=1}^{r} w1_{ij} p_j + b1_i),\ i = 1, 2, \ldots, s1 \tag{1.16}$$

(2) 输出层第 k 个神经元的输出为

$$a2_k = f2(\sum_{i=1}^{s1} w2_{ki} a1_i + b2_k),\ k = 1, 2, \ldots, s2 \tag{1.17}$$

(3) 定义误差函数为

$$E(W, B) = \frac{1}{2}\sum_{k=1}^{s2}(t_k - a2_k)^2 \tag{1.18}$$

2) 利用梯度下降法求权值变化及误差的反向传播

(1) 输出层的权值变化

对从第 i 个输入到第 k 个输出的权值有：

$$\begin{aligned}\Delta w2_{ki} &= -\eta\frac{\partial E}{\partial w2_{ki}} = -\eta\frac{\partial E}{\partial a2_k}\cdot\frac{\partial a2_k}{\partial w2_{ki}} \\ &= \eta(t_k - a2_k)\cdot f2'\cdot a1_i = \eta\cdot\delta_{ki}\cdot a1_i\end{aligned} \tag{1.19}$$

其中：

$$\delta_{ki} = f2'(t_k - a2_k)\cdot f2' = e_k\cdot f2' \tag{1.20}$$

$$e_k = t_k - a2_k \tag{1.21}$$

同理可得：

$$\Delta b2_{ki} = -\eta\frac{\partial E}{\partial b2_{ki}} = -\eta\frac{\partial E}{\partial a2_k}\cdot\frac{\partial a2_k}{\partial w2_{ki}} = \eta(t_k - a2_k)\cdot f2' = \eta\cdot\delta_{ki} \tag{1.22}$$

(2) 隐含层权值变化

对从第 j 个输入到第 i 个输出的权值，有：

$$\begin{aligned}\Delta w1_{ij} &= -\eta\frac{\partial E}{\partial w1_{ij}} = -\eta\frac{\partial E}{\partial a2_k}\cdot\frac{\partial a2_k}{\partial a1_i}\cdot\frac{\partial a1_i}{\partial w1_{ij}} \\ &= \eta\sum_{k=1}^{s2}(t_k - a2_k)\cdot f2'\cdot w2_{ki}\cdot f1'\cdot p_j \\ &= \eta\cdot\delta_{ij}\cdot p_j\end{aligned} \tag{1.23}$$

其中：

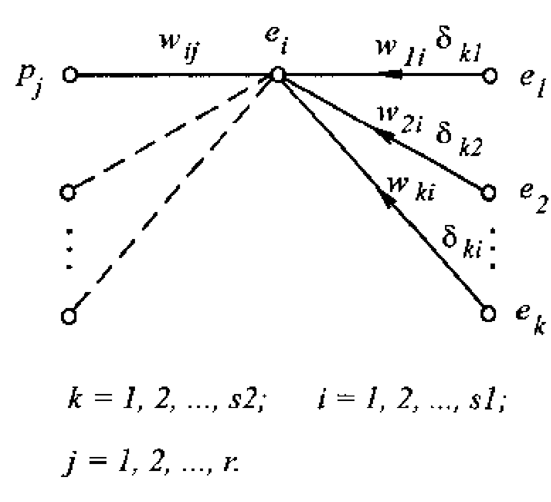
$$\delta_{ij} = e_i\cdot f1',\quad e_i = \sum_{k=1}^{s2}\delta_{ki} w2_{ki} \tag{1.24}$$

同理可得：

$$\Delta b1_i = \eta \delta_{ij} \tag{1.22}$$

3）误差反向传播的流程图与图形解释

误差反向传播过程实际上是通过计算输出层的误差 e_k，然后将其与输出层激活函数的一阶导数 $f2'$ 相乘来求得 δ_{ki}。由于隐含层中没有直接给出目标矢量，所以利用输出层的 δ_{ki} 进行误差反向传递来求出隐含层权值的变化量 $\Delta w2_{ki}$。然后计算 $e_i = \sum_{k=1}^{s2} \delta_{ki} \cdot w2_{ki}$，并同样通过将 e_i 与该层激活函数的一阶导数 $f1'$ 相乘，而求得 δ_{ij}，以此求出前层权值的变化量 $\Delta w1_{ij}$。如果前面还有隐含层，沿用上述同样方法依此类推，一直将输出误差 e_k 一层一层的反推算到第一层为止。图 1.16 给出了形象的解释。



$k = 1, 2, ..., s2;\quad i = 1, 2, ..., s1;$

$j = 1, 2, ..., r.$

图 1.16 误差反向传播法的图形解释

1.3.3 BP 网络的设计

在进行 BP 网络的设计时，一般应从网络的层数、每层中的神经元个数和激活函数、初始值以及学习速率等几个方面来进行考虑。下面讨论一下各自选取的原则。

1）网络的层数

理论上已经证明：具有偏差和至少一个 S 型隐含层加上一个线性输出层的网络，能够逼近任何有理函数。这实际上已经给了我们一个基本的设计 BP 网络的原则。增加层数主要可以更进一步地降低误差，提高精度，但同时也使网络复杂化，从而增加了网络权值的训练时间。而误差精度的提高实际上也可以通过增加隐含层中的神经元数目来获得，其训练效果也比增加层数更容易观察和调整。所以一般情况下，应优先考虑增加隐含层中的神经元数。

另外还有一个问题：能不能仅用具有非线性激活函数的单层网络来解决问题呢？结论是：没有必要或效果不好。因为能用单层非线性网络完美解决的问题，用自适应线性网络一定也能解决，而且自适应线性网络的运算速度还更快。而对于只能用非线性函数解决的问题，单层精度又不够高，也只有增加层数才能达到期望的结果。这主要还是因为一层网

络的神经元数被所要解决的问题本身限制造成的。对于一般可用一层解决的问题，应当首先考虑用感知器，或自适应线性网络来解决，而不采用非线性网络，因为单层不能发挥出非线性激活函数的特长。

输入神经元数可以根据需要求解的问题和数据所表示的方式来确定。如果输入的是电压波形，那么可根据电压波形的采样点数来决定输入神经元的个数，也可以用一个神经元，使输入样本为采样的时间序列。如果输入为图像，则输入可以用图像的像素，也可以为经过处理后的图像特征来确定其神经元个数。总之问题确定后，输入与输出层的神经元数就随之定了。在设计中应当注意尽可能地减少网络模型的规模，以便减少网络的训练时间。

2) *隐含层的神经元数*

网络训练精度的提高，可以通过采用一个隐含层，而增加其神经元数的方法来获得。这在结构实现上，要比增加更多的隐含层要简单得多。那么究竟选取多少个隐含层节点才合适？这在理论上并没有一个明确的规定。在具体设计时，比较实际的做法是通过对不同神经元数进行训练对比，然后适当的加上一点余量。

3) *初始权值的选取*

由于系统是非线性的，初始值对于学习是否达到局部最小、是否能够收敛以及训练时间的长短的关系很大。如果初始权值太大，使得加权后的输入和 n 落在了 S 型激活函数的饱和区，从而导致其导数 $f'(s)$ 非常小，而在计算权值修正公式中，因为 $\delta \propto f'(n)$，当 $f'(n) \to 0$ 时，则有 $\delta \to 0$。这使得 $\Delta w_{ij} \to 0$，从而使得调节过程几乎停顿下来。所以，一般总是希望经过初始加权后的每个神经元的输出值都接近于零，这样可以保证每个神经元的权值都能够在它们的 S 型激活函数变化最大之处进行调节。所以，一般取初始权值在(-1,1)之间的随机数。另外，为了防止上述现象的发生，威得罗等人在分析了两层网络是如何对一个函数进行训练后，提出一种选定初始权值的策略：选择权值的量级为 $\sqrt[r]{s1}$，其中 $s1$ 为第一层神经元数目。利用他们的方法可以在较少的训练次数下得到满意的训练结果，其方法仅需要使用在第一隐含层的初始值的选取上，后面层的初始值仍然采用随机取数。

4) *学习速率*

学习速率决定每一次循环训练中所产生的权值变化量。大的学习速率可能导致系统的不稳定；但小的学习速率导致较长的训练时间，可能收敛很慢，不过能保证网络的误差值不跳出误差表面的低谷而最终趋于最小误差值。所以在一般情况下，倾向于选取较小的学习速率以保证系统的稳定性。学习速率的选取范围在 0.01～0.8 之间。

和初始权值的选取过程一样，在一个神经网络的设计过程中。网络要经过几个不同的学习速率的训练，通过观察每一次训练后的误差平方和 $\sum e^2$ 的下降速率来判断所选定的学习速率是否合适。如果 $\sum e^2$ 下降很快，则说明学习速率合适，若 $\sum e^2$ 出现振荡现象，则说明学习速率过大。对于每一个具体网络都存在一个合适的学习速率。但对于较复杂网

络，在误差曲面的不同部位可能需要不同的学习速率。为了减少寻找学习速率的训练次数以及训练时间，比较合适的方法是采用变化的自适应学习速率，使网络的训练在不同的阶段自动设置不同学习速率的大小。这一方法将在后面讨论。

5）期望误差的选取

在设计网络的训练过程中，期望误差值也应当通过对比训练后确定一个合适的值，这个所谓的“合适”，是相对于所需要的隐含层的节点数来确定，因为较小的期望误差值是要靠增加隐含层的节点，以及训练时间来获得的。一般情况下，作为对比，可以同时对两个不同期望误差值的网络进行训练，最后通过综合因素的考虑来确定采用其中一个网络。

1.3.4 BP网络的限制与不足

虽然反向传播法得到广泛的应用，但它也存在自身的限制与不足，其主要表现在它的训练过程的不确定上。具体说明如下：

1）需要较长的训练时间

对于一些复杂的问题，BP 算法可能要进行几小时甚至更长的时间的训练。这主要是由于学习速率太小所造成的。可采用变化的学习速率或自适应的学习速率来加以改进。

2）完全不能训练

这主要表现在网络出现的麻痹现象上。在网络的训练过程中，当其权值调得过大，可能使得所有的或大部分神经元的加权总和偏大，这使得激活函数的输入工作在 S 型转移函数的饱和区，导致其导数$f'(s)$非常小，从而使得对网络权值的调节过程几乎停顿下来。通常为了避免这种现象的发生，一是选取较小的初始权值，另外，采用较小的学习速率，但这又增加了训练时间。

3）局部极小值

BP 算法可以使网络权值收敛到一个解，但它并不能保证所求为误差超平面的全局最小解，很可能是一个局部极小解。这是因为 BP 算法采用的是梯度下降法，训练是从某一起始点沿误差函数的斜面逐渐达到误差的最小值。对于复杂的网络，其误差函数为多维空间的曲面，就像一个碗，其碗底是最小值点。但是这个碗的表面是凹凸不平的，因而在对其训练过程中，可能陷入某一小谷区，而这一小谷区产生的是一个局部极小值。由此点向各方向变化均使误差增加，以致于使训练无法逃出这一局部极小值。

解决 BP 网络的训练问题还需要从训练算法上下工夫。在下一章中，将介绍几种有效的训练 BP 网络的快速算法。

第 2 章 网络训练优化算法

对于前向神经网络的训练，可以等价地转化为以下求极值的问题：在一个 r 维输入空间 $X=[x_1\ x_2\ \cdots x_r]^{\mathrm{T}}$，对于所定义的目标函数 $f(X)$ 求其极小值，之所以采用目标函数而不使用误差函数是因为有时性能指标中包含有误差之外的项，并且可能较复杂。由于 $f(X)$ 的复杂性，通常求助于迭代算法以有效地搜索输入空间。在迭代下降法中，下一点的 $X^{(k+1)}$ 由当前点 $X^{(k)}$ 沿方向矢量 $S(X^{(k)})$ 前进一步而确定，即可由(2.1)式统一表示：

$$X^{(k+1)}=X^{(k)}+\eta^{(k)}S(X^{(k)}) \tag{2.1}$$

其中，$\eta^{(k)}$ 是正的步长，控制沿该方向增加的数量。在神经网络的研究中，称其为学习速率；k 表示当前的迭代次数。

迭代下降法一般分两步计算第 k 步的 $\eta^{(k)}S(X^{(k)})$：首先确定方向 $\eta^{(k)}$，随后计算步长 $S(X^{(k)})$。而第 $k+1$ 步的计算应满足以下不等式：

$$f(X^{(k+1)})=f(X^{(k)})+\eta^{(k)}S(X^{(k)})<f(x^{(k)}) \tag{2.2}$$

在一般对梯度算法的不同改进方法中，基本差异在于以第 k 步确定第 $k+1$ 步的方向上。一旦确定了方向，所有算法都是朝着由 $X^{(k)}$ 和方向 $S(X^{(k)})$ 所确定直线上的极小点移动。如果直线下降方向 $S(X^{(k)})$ 是在目标函数的梯度基础上确定的，那么这种下降法就称为梯度下降法。

虽然反向传播法得到广泛的应用，但由于采用的是梯度下降法，所以它存在着自身的限制与不足，其主要表现在其训练时间过长并易于陷入局部极小值。实际上，传统的基于标准梯度下降法的 BP 算法在求解实际问题时常因收敛速度太慢而影响求解质量。因此，80 年代中期以来，许多研究人员对其做了深入的研究，人们在标准 BP 算法的基础上，进行了许多有益的改进，主要目标是为了加快训练速度，避免陷入局部极小值和改善其它能力。另外，还提出了不少基于非线性优化的训练算法，可使网络训练的收敛速度比标准梯度下降法快数十乃至数百倍。

本章试图通过对几种具有代表性的改进算法性能上的对比，从实用的角度提醒人们注意这样一个事实，那就是：除了基于标准梯度下降法可以进行算法改进外，基于数值优化方法的改进算法可能得到意想不到的惊人的收敛速度。因此，我们将改进算法分为两大类：第一类为基于标准梯度下降的改进方法，它们是由标准梯度下降法发展而来，其中，选用的有附加动量的 BP 算法、可变学习速率的 BP 算法和弹性 BP 算法。第二类方法为基于标准数值优化的改进方法，其中，选用共轭梯度法、拟牛顿法和 Levenberg-Marquardt 法。

下面，首先给出各种算法的计算公式及其工作原理，并从理论上分析各自的优缺点，然后用两个实例详细进行性能的对比分析，最后给出小结。

2.1 基于标准梯度下降的方法

标准的 BP 算法是基于梯度下降法，通过计算目标函数对网络权值及其偏差的梯度对其进行修正的，改进的算法都是在标准的梯度下降法的基础上发展而来的，它们均只用到目标函数对权值和偏差的一阶导数（梯度）信息。

标准梯度下降法权值和偏置值修正的迭代过程可以表示为：

$$X^{(k+1)} = X^{(k)} - \eta \nabla f(X^{(k)}) \tag{2.3}$$

其中，$X^{(k)}$ 为由网络的所有的权值和偏置值组成的向量；η 为学习速率；$f(X^{(k)})$ 为目标函数，$\nabla f(X^{(k)})$ 表示目标函数的梯度。

标准 BP 算法虽然为训练网络提供了简单有效的方法，但由于在训练过程中学习速率 η 为一个较小的常数，因而使它存在着收敛速度慢以及局部极小问题。为了解决这些问题，人们提出了许多改进算法。其中，比较有代表性的有以下几种。

2.1.1 附加动量法

附加动量法使网络在修正其权值时，不仅考虑误差在梯度上的作用，而且考虑在误差曲面上变化趋势的影响，其作用如同一个低通滤波器，它允许网络忽略网络上的微小变化特性。在没有附加动量的作用下，网络可能陷入浅的局部极小值，利用附加动量的作用则有可能滑过这些极小值。

该方法是在反向传播法的基础上在每一个权值的变化上加上一项正比于前次权值变化量的值，并根据反向传播法来产生新的权值变化。附加动量的 BP 算法的权值修正的迭代过程可以表示为：

$$\Delta X^{(k+1)} = mc * \Delta X^{(k)} - (1-mc) * \eta \nabla f(X^{(k)}) \tag{2.4}$$

其中，k 为训练次数，mc 为动量因子，一般取 0.95 左右。

附加动量法的实质是将最后一次权值变化的影响，通过一个动量因子来传递。当动量因子取值为零时，权值的变化仅是根据梯度下降法产生；当动量因子取值为 1 时，新的权值变化则是设置为最后一次权值的变化，而依梯度法产生的变化部分则被忽略掉了。以此方式，当增加了动量项后，促使权值的调节向着误差曲面底部的平均方向变化，当网络权值进入误差曲面底部的平坦区时，梯度将变得很小，于是，$\Delta X^{(k+1)} \approx \Delta X^{(k)}$，从而防止了 $\Delta X^{(k+1)} = 0$ 的出现，有助于使网络从误差曲面的局部极小值中跳出。

根据附加动量法的设计原则，当修正的权值在误差中导致太大的增长结果时，新的权值应被取消而不被采用，并使动量作用停止下来，以使网络不进入较大误差曲面；当新的误差变化率对其旧值超过一个事先设定的最大误差变化率时，也得取消所计算的权值变

化。其最大误差变化率可以是任何大于或等于1的值。典型的值取1.04。所以在进行附加动量法的训练程序设计时，必须加进条件判断，以正确使用其权值修正公式。

训练程序中对采用动量法的判断条件为：

$$mc=\begin{cases}0 & SSE^{(k)}>SSE^{(k-1)}\cdot 1.04\\ 0.95 & SSE^{(k)}<SSE^{(k-1)}\\ mc & \text{其他}\end{cases} \tag{2.5}$$

其中，SSE 为网络的输出误差平方和。

在附加动量的作用下，当网络的训练误差落入局部极小值后，能够产生一个继续向前的正向斜率的运动，并跳出较浅的峰值，落入全局最小值。然后，仍然在附加动量的作用下，达到一定的高度后（即产生了一个 $SSE^{(k)}> SSE^{(k-1)}*1.04$）自动返回，并像弹子滚动一样来回摆动，直至停留在最小值点上。

通过实际应用我们发现，训练参数的选择对采用动量法的网络训练效果的影响是相当大的。如学习速率太大，将导致其误差值来回振荡；学习速率太小，则导致太小的动量能量，从而使其只能跳出很浅的"坑"。对于较大的"坑"或"谷"将无能为力。而从另一方面来看，其误差相对于权值的曲线（面）的形状与凹凸性是由问题的本身决定的，所以每个问题都是不相同的。这必然对学习速率的选择带来了困难。一般情况下只能采用不同的学习速率进行对比（典型值取 0.05）。另外，对于这种网络的训练必须给予足够的训练次数，以使其训练结果为最后稳定到最小值时得到的结果，而不是得到一个正好摆动到较大误差值时的网络权值。

此训练方法也存在缺点。它对训练的初始值有要求，必须使其值在误差曲线上的位置所处误差下降方向与误差最小值的运动方向一致。如果初始误差点的斜率下降方向与通向最小值的方向背道而驰，则附加动量法失效。训练结果将同样落入局部极小值而不能自拔。初始值选得太靠近局部极小值时也不行，所以建议多用几个初始值先粗略训练几次以找到合适的初始位置。另外，学习速率太小时，网络也没有足够的能量跳过低"谷"。

2.1.2 自适应学习速率

对于一个特定的问题，要选择适当的学习速率不是一件容易的事情。通常是凭经验或实验获取，但即使这样，对训练开始初期功效较好的学习速率，不见得对后来的训练合适。为了解决这一问题，人们自然会想到使网络在训练过程中自动调整学习速率。通常调节学习速率的准则是：检查权值的修正值是否真正降低了误差函数，如果确实如此，则说明所选取的学习速率值小了，可以对其增加一个量；若不是这样，而产生了过调，那么就应该减小学习速率的值。与采用附加动量法时的判断条件相仿，当新误差超过旧误差一定的倍数时，学习速率将减少；否则其学习速率保持不变；当新误差小于旧误差时，学习速率将被增加。此方法可以保证网络总是以最大的可接受的学习速率进行训练。当一个较大的学习速率仍能够使网络稳定学习，使其误差继续下降，则增加学习速率，使其以更大的学习速率进行学习。一旦学习速率调得过大，而不能保证误差继续减少，则减少学习速率直到

使其学习过程稳定为止。下式给出了一种自适应学习速率的调整公式：

$$\eta(k+1)=\begin{cases}1.05\eta^{(k)} & SSE^{(k)}<SSE^{(k-1)}\\0.7\eta^{(k)} & SSE^{(k)}>1.04\cdot SSE^{(k-1)}\\\eta(k) & \text{其他}\end{cases} \tag{2.6}$$

同样，*SEE* 为网络输出误差和。初始学习速率 $\eta^{(0)}$ 的选取范围可以有很大的随意性。实践证明采用自适应学习速率的网络训次数只是固定学习速率网络训练次数的几十分之一。所以具有自适应学习速率的网络训练是极有效的训练方法。

在 BP 网络的实际设计和训练中，总是要加进上述两种改进方法中的一种，或两种同时应用，人们几乎不再采用单纯的 BP 算法。虽然这些改进算法具有较多的参数变化，但在许多常用的应用软件包中，都已有现成的改进算法可以直接调用，从而对于神经网络的设计者来说，可以把更多的精力用到掌握和了解更多和更好的改进算法上，以选择适当的算法来训练网络。

2.1.3 弹性 BP 算法

BP 网络通常采用 Sigmoid 隐含层。Sigmoid 函数常被称为“压扁”函数，它将一个无限的输入范围压缩到一个有限的输出范围。其特点是当输入很大时，斜率接近 0，这将导致算法中的梯度幅值很小，可能使得对网络权值的修正过程几乎停顿下来。

弹性 BP 算法只取偏导数的符号，而不考虑偏导数的幅值。偏导数的符号决定权值更新的方向，而权值变化的大小由一个独立的“更新值”确定。若在两次连续的迭代中，目标函数对某个权值的偏导数的符号不变号，则增大相应的“更新值”（如在前一次“更新值”的基础上乘 1.3）；若变号，则减小相应的“更新值”（如在前一次“更新值”的基础上乘 0.5）。其权值修正的迭代过程可表示如下：

$$X^{(k+1)}=X^{(k)}-\Delta X^{(k)}*\mathrm{sign}(\nabla fX^{(k)}) \tag{2.7}$$

其中，$\Delta X^{(k)}$ 为前一次的“更新值”，其初始值 $\Delta X^{(k)}$ 要根据实际应用预先设定。

在弹性 BP 算法中，当训练发生振荡时，权值的变化量将减小；当在几次迭代过程中权值均朝一个方向变化时，权值的变化量将增大。因此，一般来说，弹性 BP 算法的收敛速度要比前述几种方法快得多。而且算法并不复杂，也不需要消耗更多的内存。

以上三种改进算法的存储量要求相差不大，各算法的收敛速度依次加快。其中，弹性 BP 算法的收敛速度远快于前两者。大量实际应用已证明弹性 BP 算法非常有效。

因此，在实际应用的网络训练中，当采用附加动量法乃至可变学习速率的 BP 算法达不到训练要求时，可以采用弹性 BP 算法或下面介绍的基于数值优化的方法。

2.2 基于数值优化方法的网络训练算法

上一节所介绍的几种基于一阶梯度的方法用于简单问题时往往可以很快地收敛到期望

值，然而，当用于较复杂的实际问题时，除了弹性 BP 算法以外，其余算法在收敛速度上都存在着一定的问题。

BP 网络的训练实质上是一个非线性目标函数的优化问题，人们对非线性优化问题的研究已有数百年的历史，而且，不少传统数值优化方法收敛也较快。因而，人们自然想到采用基于数值优化方法的算法对 BP 网络的权值进行训练。与梯度下降法不同，基于数值优化的算法不仅利用了目标函数的一阶导数信息，往往还利用目标函数的二阶导数信息。这类算法，包括拟牛顿法、Levenberg-Marquardt 法和共轭梯度法，它们可以统一描述为：

$$
\begin{aligned}
f(X^{(k+1)}) &= \min_{\eta} f(X^{(k)} + \eta^{(k)} S(X^{(k)})) \\
X^{(k+1)} &= X^{(k)} + \eta^{(k)} S(X^{(k)})
\end{aligned}
\tag{2.8}
$$

其中，$X^{(k)}$ 为网络所有的权值和偏置值组成的向量，$S(X^{(k)})$ 为由 X 的各分量组成的向量空间中的搜索方向，$\eta^{(k)}$ 为在 $S(X^{(k)})$ 的方向上，使 $f(X^{(k+1)})$ 达到极小的步长。这样，网络权值的寻优分为两步：首先，确定当前迭代的最佳搜索方向 $S(X^{(k)})$；而后，在此方向上寻求最优迭代步长。关于最优搜索步长 $\eta^{(k)}$ 的选取，是一个一维搜索（线搜索）问题。对这一问题有许多方法可供选择，如黄金分割法、二分法、多项式插值、回溯法等。以下所讨论的三种方法的区别正在于对最佳搜索方向 $S(X^{(k)})$ 的选择上有所不同。

2.2.1 拟牛顿法

牛顿法是一种常见的快速优化方法，它利用了一阶和二阶导数信息，其基本形式是：

第一次迭代的搜索方向确定为负梯度方向，即搜索方向 $S(X^{(0)}) = -\nabla f(X^{(0)})$，以后各次迭代的搜索方向由下式确定：

$$S(X^{(k)}) = -(H^{(k)})^{-1} \nabla f(X^{(k)}) \tag{2.9}$$

即

$$X^{(k+1)} = X^{(k)} - \eta^{(k)} S(X^{(k)}) = X^{(k)} - \eta^{(k)} (H^{(k)})^{-1} \nabla f(X^{(k)}) \tag{2.10}$$

其中，$H^{(k)}$ 为海森（Hessian）矩阵（二阶导数矩阵）。牛顿法的收敛速度比一阶梯度法快，不过由于神经网络中参数数目的庞大，导致计算海森矩阵的复杂性增加。

因此，人们在牛顿法的基础上，提出了一类无需计算二阶导数矩阵及其求逆运算的方法。这类方法一般是利用梯度信息或一个近似矩阵去逼近 $H^{(k)}$。不同的构造 $H^{(k)}$ 的方法，就产生了不同的拟牛顿法。显然，拟牛顿法是为了克服梯度下降法收敛慢以及牛顿法计算复杂而提出的一种算法。下面，介绍两种比较典型的拟牛顿法：BFGS 拟牛顿法和正割拟牛顿法。

1）BFGS 拟牛顿法

除了第一次迭代外，对应（2.9）、（2.10）两式，BFGS 拟牛顿法在每一次迭代中采用下式来逼近海森矩阵：

$$H^{(k)} = H^{(k-1)} + \frac{\nabla f(X^{(k-1)}) * \nabla f(X^{(k-1)})^T}{\nabla f(X^{(k-1)})^T * S(X^{(k-1)})} + \frac{\mathrm{dg}X * \mathrm{dg}X^T}{\mathrm{dg}X^T * \eta^{(k-1)} S(X^{(k-1)})} \tag{2.11}$$

其中，$\mathrm{dg}X = \nabla f(X^{(k)}) - \nabla f(X^{(k-1)})$

BFGS 拟牛顿法在每次迭代中都要存储近似的海森矩阵，海森矩阵是一个 $n*n$ 的矩阵，n 是网络中所有的权值和偏置的总数。因此，当网络参数很多时，要求极大的存储量，计算也较为复杂。

2）正割拟牛顿法

正割拟牛顿法不需要存储完整的海森矩阵，除了第一次迭代外，以后各次迭代的搜索方向由下式确定

$$S(X^{(k)}) = -\nabla f(X^{(k)}) + A_c * \eta^{(k-1)} S(X^{(k-1)}) + B_c * \mathrm{dg}X \tag{2.12}$$

其中

$$\mathrm{dg}X = \nabla f(X^{(k)}) - \nabla f(X^{(k-1)})$$

$$B_c = \frac{S(X^{(k-1)}) * \nabla f(X^{(k)})}{S(X^{(k-1)})^T * \mathrm{dg}X}$$

$$A_c = -\left(1 + \frac{\mathrm{dg}X^T * \mathrm{dg}X}{S(X^{(k-1)})^T * \mathrm{dg}X}\right) * B_c + \frac{\mathrm{dg}X^T * \nabla f(X^{(k)})}{S(X^{(k-1)})^T * \mathrm{dg}X}$$

相对于 BFGS 拟牛顿法，正割拟牛顿法减小了存储量与计算量。实际上，正割拟牛顿法是通常的需要近似计算海森矩阵的拟牛顿法与后面要介绍的共轭梯度法的一种折衷，它的形式与共轭梯度法相似。

2.2.2 共轭梯度法

鉴于梯度下降法收敛速度慢，而拟牛顿法计算较复杂，共轭梯度法力图避免两者的缺点。共轭梯度法的第一步沿负梯度方向进行搜索，然后沿当前搜索方向的共轭方向进行搜索，可以迅速达到最优值。其过程描述如下：

第一次迭代的搜索方向确定为负梯度方向，即搜索方向 $S(X^{(0)}) = -\nabla f(X^{(0)})$，以后各次迭代的搜索方向由下式确定：

$$S(X^{(k)}) = -\nabla f(X^{(k)}) + \beta^{(k)} S(X^{(k-1)}) \tag{2.13}$$

其中，$X^{(k)} = X^{(k)} + \eta^{(k)} S(X^{(k)})$ 。

根据 $\beta^{(k)}$ 所取形式的不同，可构成不同的共轭梯度法。常用的两种形式是：

$$\beta^{(k)}=\frac{g_k^{\mathrm{T}}g_k}{g_{k-1}^{\mathrm{T}}g_{k-1}} \text{或} \beta^{(k)}=\frac{\Delta g_{k-1}^{\mathrm{T}}g_k}{g_{k-1}^{\mathrm{T}}g_{k-1}} \tag{2.14}$$

其中，$g_k=\nabla f(X^{(k)})$。

通常，搜索方向 $S(X^{(k)})$ 在迭代过程中以一定的周期复位到负梯度方向，周期一般为 n（网络中所有的权值和偏差的总数目）。

共轭梯度法比绝大多数常规的梯度下降法收敛都要快，而且只需增加很少的存储量及计算量。因而，对于权值很多的网络采用共轭梯度法不失为一个较好的选择。

2.2.3 Levenberg-Marquardt 法

Levenberg-Marquardt 法实际上是梯度下降法和牛顿法的结合。我们知道，梯度下降法在开始几步下降较快，但随着接近最优值时，由于梯度趋于零，使得目标函数下降缓慢；而牛顿法可以在最优值附近产生一个理想的搜索方向。Levenberg-Marquardt 法的搜索方向定为：

$$S(X^{(k)})=-(H^{(k)}+\lambda^{(k)}I)^{-1}\nabla f(X^{(k)}) \tag{2.15}$$

令 $\eta^{(k)}=1$，则 $X^{(k+1)}=X^{(k)}+S(X^{(k)})$。

起始时，λ 取一个很大的数（如10^4），此时相当于步长很小的梯度下降法；随着最优点的接近，λ 减小到零，则 $S(X^{(k)})$ 从负梯度方向转向牛顿法的方向。通常，当 $f(X^{(k+1)})<f(X^{k})$，减小 λ（如 $\lambda^{(k+1)}=0.5\lambda^{(k)}$）；否则，增大 λ（如 $\lambda^{(k+1)}=2\lambda^{(k)}$）。

从（2.15）式中可以注意到该方法仍然需要求海森矩阵。不过，由于在训练 BP 网络时目标函数常常具有平方和的形式（这也是该算法最初所要解决的问题），则海森矩阵可通过雅可比（Jacobian）矩阵进行近似计算 $H=J^TJ$，雅可比矩阵包含网络误差对权值及偏差的一阶导数，而通过标准的反向传播技术计算雅可比矩阵要比计算海森矩阵容易得多。

Levenberg-Marquardt 需要的存储量很大，因为雅可比矩阵使用一个 $Q*n$ 矩阵，Q 是训练样本的个数，n 是网络中所有的权值和偏差的总数目。为此，也产生了其他一些用以降低内存的 Levenberg-Marquardt 法。

Levenberg-Marquardt 的长处是在网络权值数目较少时收敛非常迅速。

综上所述，显然，Levenberg-Marquardt 法和拟牛顿法因为要近似计算海森矩阵，需要较大的存储量，不过，通常这两类方法的收敛速度较快。其中，Levenberg-Marquardt 法结合了梯度下降法和牛顿法的优点，性能更加优良一些。共轭梯度法所需存储量较小，但收敛速度相对前两种方法慢。所以，考虑到网络参数的数目（即网络中所有的权值和偏差的总数目），在选择算法对网络进行训练时，可以遵照以下原则：

①在网络参数很少时，可以使用牛顿法或 Levenberg-Marquardt 法；

②在网络参数适中时，可以使用拟牛顿法；

③在网络参数很多，需要考虑到存储容量问题时，不妨选择共轭梯度法。

其实，对于不同的问题，很难比较算法的优劣。而对于特定问题，一种通常较好的方法有时却可能不易获得良好的训练效果，甚至可能出现难于收敛到预定目标的情况。因此，在解决实际问题时，应当尝试采用多种不同类型的训练算法，以期获得满意的结果。在大多数情况下，可以使用 Levenberg-Marquardt 法和拟牛顿法。此外，弹性 BP 算法也是一种简单有效的方法。

需要指出的是，本章述及的所有算法均存在局部极小问题，就经验而言，对大多数问题，Levenberg-Marquardt 法可以获得相对较好的结果，然而这一结论没有任何理论依据，所以对网络应当使用不同的初始值进行多次的训练。此外，当然还可以采用其他全局优化算法，如模拟退火法、遗传算法等，来解决局部极小问题。

2.3 数值实例对比

本节将对上述的所有算法的实际应用效果进行数值实验对比分析。虽然有些算法的实现较为复杂，不过在计算机软件的辅助下已经不成问题，尤其在控制界常用的 MATLAB 5.3 的神经网络工具箱中，已有实现上述训练算法的函数，设计者只需进行适当的调用即可对网络进行训练。

本节中将给出两个例子，均在 MATLAB 5.3 环境下进行。第一个例子为利用神经网络的非线性逼近的特性对一个非线性函数对象进行逼近；第二个例子是利用神经网络对一个真实的非线性直流电机进行建模。

2.3.1 非线性函数的逼近

构造非线性函数对象：

$$y(k)=y(k-1)/\left(1+y^2(k-1)\right)+u^3(k-1) \tag{2.16}$$

训练用输入信号取为：

$$u(k)=0.2\sin(2\pi k/25)+0.3\sin(\pi k/15)+0.3\sin(\pi k/75) \tag{2.17}$$

将信号（2.17）式作为非线性对象（2.16）式的输入信号，取 k 从 0 到 200 的输入/输出作为训练样本，网络训练样本的输入/输出关系如图 2.1 所示。

在 MATLAB 5.3 中，如果要一个网络完成某种功能，设计者可以通过调用函数 newff 实现网络的创建，然后再调用函数 train 对所建网络 newff 进行训练。

函数 newff 带有四个参数：对于一个输入节点数为 R 的网络，第一个参数是一个 R*2 的矩阵，其中，第 i 行的两个元素依次为所有输入样本中对应于第 i 个输入的最小与最大范围；第二个参数是一个给出各层神经元个数的数组；第三个参数是各层采用的激活函数

的名称；第四个输入参数是所选用的网络训练算法的名称。newff 的返回值是一个经过初始化后的网络对象。本例中，用于对非线性函数的逼近的网络可创建如下：

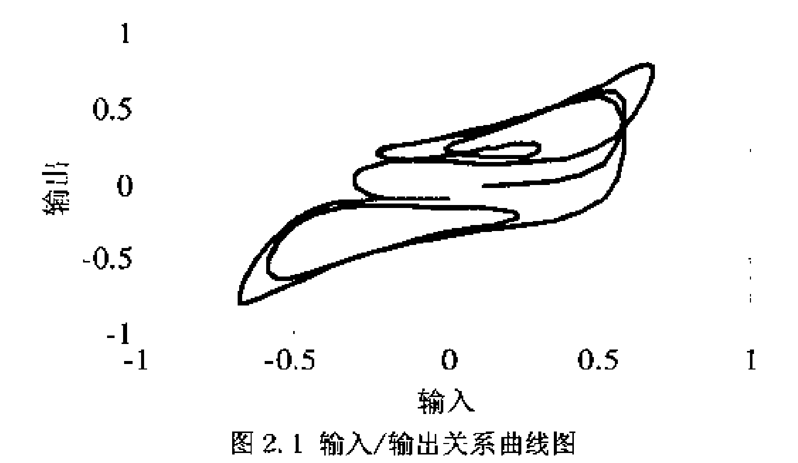


图 2.1 输入/输出关系曲线图

$$
\begin{aligned}
&p = [u; y_d] \\
&net = newff(minmax(p),[5,1],\{'tansig','purelin'\},'traingd')
\end{aligned}
\quad (2.18)
$$

在神经网络工具箱中，前述各种训练算法已经编制成.m 文件，设计者只要在 newff 的第四个参数中选用希望采用的算法名称，就可调用函数 train 进行网络的训练。函数 train 有三个参数：第一个参数是由 newff 创建的网络；第二个参数是输入矩阵；第三个参数是期望输出矩阵。

$$net = train(net,p,y) \quad (2.19)$$

在（2.18）、（2.19）式中，u 为所有 200 个样本输入组成的行向量；y_d 为 200 个样本输出 y 经一阶延时组成的行向量。即网络的输入向量有两个，u 与 y_d，它们组成输入矩阵 p；网络的期望输出为 y（200 个样本输出组成的行向量）。Minmax(p)意为取得输入向量的最小到最大的范围；网络的第一层（即隐含层）包含有五个节点，第二层（即输出层）包含有一个节点。隐含层采用 tan-Sigmoid 函数；输出层采用线性函数 purelin。所选训练算法 traingd 为标准的梯度下降法。

作为对比，我们始终采用（2.18）式中的网络结构，而只改变最后的算法参数，采用（2.19）式对网络进行训练，以此来观察各种训练算法的收敛速度。

值得注意的是，开始训练之前，必须对网络的权值与偏差进行初始化。好在函数newff自动调用缺省，这只是一种初始化方法。

我们把标准梯度下降法、附加动量的BP算法、学习速率可变的BP算法放在一起进行比较。训练中，以固定最大迭代次数20000为标准。网络的初始化采用Nguyen与Widrow提出的名为initnw的初始化函数，它可以使每一层神经元的激活区域大致上均匀分布在输入空间上。考虑到每次对网络的权值与偏置值的初始化结果不完全相同，因此对每种算法应进行多次训练，表2.1给出了三种算法收敛速度的比较。表中的数据均为五次训练的平均值。

由表2.1可见，三种方法所用时间都在400秒以上，不过，在误差的变化上，学习速率可变的BP算法的收敛速度明显比前两种快，它达到了0.0001的数量级。

表2.1 三种算法收敛速度的比较

函数	算法描述	时间(秒)	均方误差
traingda	变学习速率的BP算法	448.30	0.000265963
traingdm	附加动量的BP算法，动量因子mc=0.9	457.75	0.00219000
traingd	标准梯度下降法（标准BP算法）	446.38	0.00383469

鉴于弹性BP算法及基于数值优化的算法收敛速度很快，可用固定目标函数（均方误差）SSE = 0.00001来进行性能的对比实验，使各算法一直迭代到满足目标要求才停止。五次成功训练的平均值记录如表2.2所示。表中给出了MATLAB 5.3中所有的有四种不同的共轭梯度法和两种不同的拟牛顿法的训练结果。

对照以上两表可以看出，表2.2列出的八种算法比表2.1中的三种算法，无论在运算时间，还是在收敛精度上，都有着绝对的优势，两者不在同一个数量级上。由此可见，在标准梯度下降法基础上进行改进的算法，所取得的进步是很有限的。要想有数量级上的突破，必须转变思路。基于数值优化的方法就是一个极好的尝试。我们应当在今后的研究与应用中，采用这些现成的软件编制好的训练网络的算法，以提高网络设计的精度，灵活地应用于实际问题中。

表2.2 快速训练算法的收敛速度对比

函数	算法描述	时间(秒)	迭代次数
traincgf	Fletcher-Powell 共轭梯度法，其中 $\beta^{(k)}=\frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$	14.34	238
traincgp	Polak-Ribiere共轭梯度法，其中 $\beta^{(k)}=\frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}}$	12.32	216
traincgb	Powell-Beale共轭梯度法，当前梯度与前一梯度正交性很小时，将搜索方向复位到负梯度方向	10.11	167
trainscg	Scaled共轭梯度法，一种无需进行线搜索的方法	11.76	263
trainbfg	BFGs拟牛顿法	4.34	71
trainoss	One-Step-Secant，正割拟牛顿法	22.50	437
trainlm	Levenberg-Marquardt法	0.77	7
trainrp	弹性BP算法	50.20	2239

2.3.2 逼近非线性直流电机的输入/输出特性

下面，将再对一个实际存在的具有非线性摩擦力影响的直流电机的输入/输出特性进行神经网络建模，并采用不同的训练算法进行对比。用来采集输入/输出信号的实际控制

系统包括一台 Pentium 200 计算机、一块内置于计算机的 12 位 A/D，D/A 转换板、PWM 功率放大电路、直流力矩电动机及用于速度反馈的直流测速发电机。模拟电压输入范围与输出控制电压范围均为[-5,5]伏，模数转换后的数字量范围均为[-2048,2048]，为了方便起见，输入和输出单位均采用数字量。被控系统的模型包括除计算机外的所有部件的集合。

给电机系统输入幅值为 300，周期为 10s 的正弦信号，在 5ms 的采样周期下，获得电机的真实输出（即转速信号）。实际电机的输入/输出如图 2.2 所示。从中可见系统中存在着严重的非线性特性。

实验表明，采用标准梯度下降法、附加动量的 BP 算法、学习速率可变的 BP 算法来训练网络模型，无法达到性能目标。我们曾采用可变学习速率的 BP 算法训练了 3 个小时，训练的误差仍然离目标相差很远。

假设电机模型为一阶系统，即输入/输出关系式可以表示为

$$y(k)=f(y(k-1),u(k-1)) \tag{2.20}$$

那么我们仍然可用上例中的网络结构 newff 来训练网络模型，训练集样本数为 2000 个。

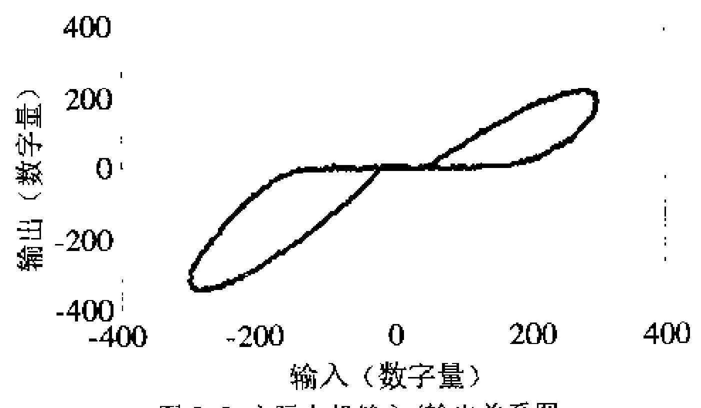


图 2.2 实际电机输入/输出关系图

经过多次实验，除 trainscg 共轭梯度法外，其余三种共轭梯度法均在离目标较远的点陷入局部极小值，使得线搜索步长趋近于零而终止训练。trainscg 法虽然没有陷入局部极小值，但收敛也很慢。曾经进行了 100000 次迭代，耗时近三个小时，均方误差 *SSE* 才达到 22.8295(考虑到输出信号的幅值,这个结果勉强可以接受)。拟牛顿法中的 trainoss 法收敛速度也很慢，根本无法实用。

最后，只有表2.3中的三种方法，在固定目标函数（均方误差）为*SSE* = 2的条件下，可以使迭代一直进行到满足目标要求为止。

表2.3 三种算法的比较

函数	算法描述	时间(秒)	迭代次数
trainrp	弹性BP算法	314.12	3483
trainbfg	BFGSS拟牛顿法	29.50	133
trainlm	Levenberg-Marquardt法	10.27	45

由此可见，在实际应用中，要针对具体的问题尝试多种算法来训练网络，以达到最佳

效果。另外，从两个例子中可以看出，BFGS拟牛顿法和Levenberg-Marquardt法两种算法均有上乘表现，其中，后者的收敛速度最快，在相同的误差目标下所花费的时间之短，是其他大多数算法所无法比拟的。

2.4 小　结

本章对几种训练BP网络权值的改进算法进行了性能上的分析和对比。通过对实验结果的对比分析表明，采用基于标准数值优化方法的各种改进算法要比基于标准梯度下降法的改进算法在收敛速度上有着一个数量级以上的提高。这为采用神经网络对复杂系统进行建模与控制提供了可行性。不过，这里所讨论的算法均存在陷入局部极小值的可能性。这一问题可以通过其他途径加以解决。

第 3 章 BP 网络在智能系统中的建模与控制

人工神经网络已在各个领域得到广泛的应用，尤其是在智能系统中的非线性建模及其控制器的设计、模式分类与模式识别、联想记忆和优化计算等方面更是得到人们的极大关注。从控制理论的观点来看，神经网络对非线性函数的逼近能力是最有意义的，因为非线性系统的变化多端是采用常规方法无法建模和控制的基本原因，而神经网络描述非线性图形的能力，也就是对非线性系统的建模能力，正适合于解决非线性系统建模与控制器综合中的这些问题。具有这一特性的前向反向传播网络在实际应用中的比例要占所有人工神经网络应用的 80%以上。所以下面从 BP 网络入手，介绍 BP 网络在控制系统中的非线性建模、控制器设计以及多种常用的系统控制方案。

虽然人工神经网络的种类已多达上百种，但应用最多的还是 BP 网络。不过 BP 网络所具有的基本功能只有一个，那就是非线性连续有理函数的逼近功能。对控制系统中的非线性过程的建模以及对系统的控制器的设计也是利用 BP 网络这一基本的功能。

3.1 直接正向模型建立

假定（被控）系统离散型非线性差分方程为：

$$y_p(k+1)=f(y_p(k),\dots,y_p(k-n+1),u(k),\dots,u(k-m+1)) \tag{3.1}$$

即由非线性函数 f 所确定的系统，在 $k+1$ 时刻的输出取决于过去 n 个时刻的输出值，以及过去 m 个时刻的输入值。

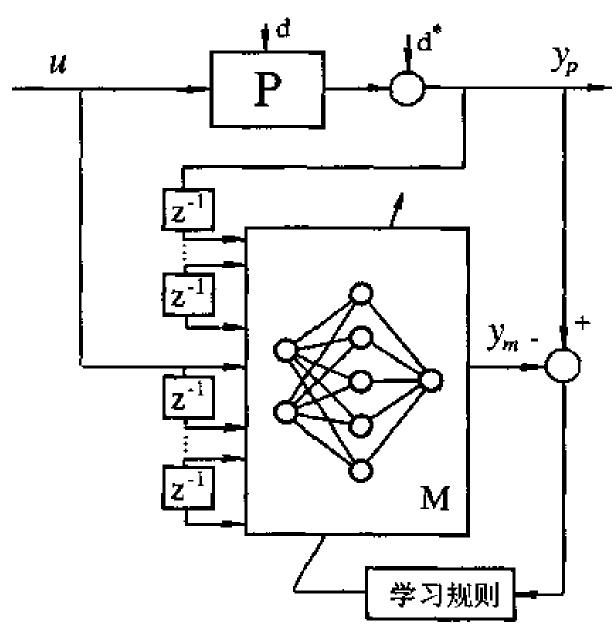


图 3.1 网络建模训练时结构图

这里我们关心的是系统响应的动态部分，模型对系统干扰的表达式是隐含在其中的。

那么对系统建模的一个直接的方法是通过选择与系统相同的输入/输出数据作为训练

用样本，对神经网络进行训练，其目的是为了使网络的输入/输出能够与系统的输入/输出完全一致。若取网络的输出变量为 y_m，那么训练网络时的网络结构图应如图 3.1 所示。

根据图 3.1 所示的网络建模训练图可以写出网络输入与输出之间的关系式为：

$$y_m(k+1)=\hat{f}(y_p(k),\ldots,y_p(k-n+1),u(k),\ldots,u(k-m+1)) \tag{3.2}$$

此处，$\hat{f}$ 代表网络的非线性输入/输出映照关系（即为函数 f 的逼近）。注意训练中的网络输入包括真实系统过去时刻的输出值（作为样本输入的一部分），不过此时网络没有反馈。

假定网络在经过适当的训练过程后，得到了满意的函数拟合（即获得 $y_m \approx y_p$），那么网络的训练则完成，并在网络以后的工作中，可以将网络自身的输出反馈回来作为网络输入的一部分，以这种方法使网络能够独立的使用，如图 3.2 所示。

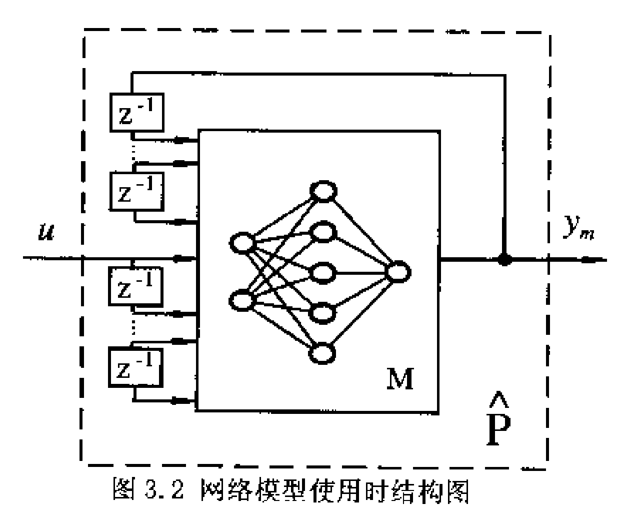


图 3.2 网络模型使用时结构图

由图 3.2 可以得到网络模型的输入/输出关系式为：

$$y_m(k+1)=\hat{f}(y_m(k),\ldots,y_m(k-n+1),u(k),\ldots,u(k-m+1)) \tag{3.3}$$

由于系统建模的需要，网络在结构上将其输出经过延时，作为网络输入的一部分。所以，控制系统中用于建模的网络可以被称为“带有反馈变量的前向网络”，它与霍普菲尔德型的反馈网络仍然是两种截然不同的网络。

3.2 逆模型建立

采用非线性系统的逆模型与系统本身相串联而消除其非线性影响的思想，促使人们感兴趣对逆模型的建立。人们最容易想到的利用神经网络建立逆模型的方法则是将（被控）系统的输入和输出数据分别作为网络的输出和输入来训练网络获得。即所谓的直接逆模型建立，其网络的训练用结构图如图 3.3 所示。从图中可以看出，与直接建模一样，网络的输入利用了系统输入/输出的综合信息，网络训练采用系统输入信号与网络输出信号的误差来修正网络权值。从这个训练网络的结构上还可以看出，训练网络的目的是迫使网络作为系统的逆模型。但是，这种方法存在着缺陷，首先，网络权值的修正过程不是采用期望的样本进行学习训练的。训练逆模型所用的输入信号起码应当遍及控制器可能输入的所有

范围。但实际训练时所采用的样本不可能被事先定义。另外，控制系统的实际目标是希望通过控制器的行为产生期望的系统输出，而训练逆模型时所产生的系统输出也并不是期望的系统输出。所以，在直接逆建模中所采用的训练样本并不与所需要的训练样本相对应，因而必然有可能存在未被训练到的控制域。其次，如果该非线性系统的映照关系不是一一对应，那么，可能产生一个不正确的逆模型。实际应用中常被人采用的建立逆模型的方法是能够克服上述缺点的第二种方法，可以被称为间接逆模型训练法。在此训练方法中，网络逆模型被置于（被控）系统的前端，并且将网络的输出作为系统的输入，如图 3.4 所示。

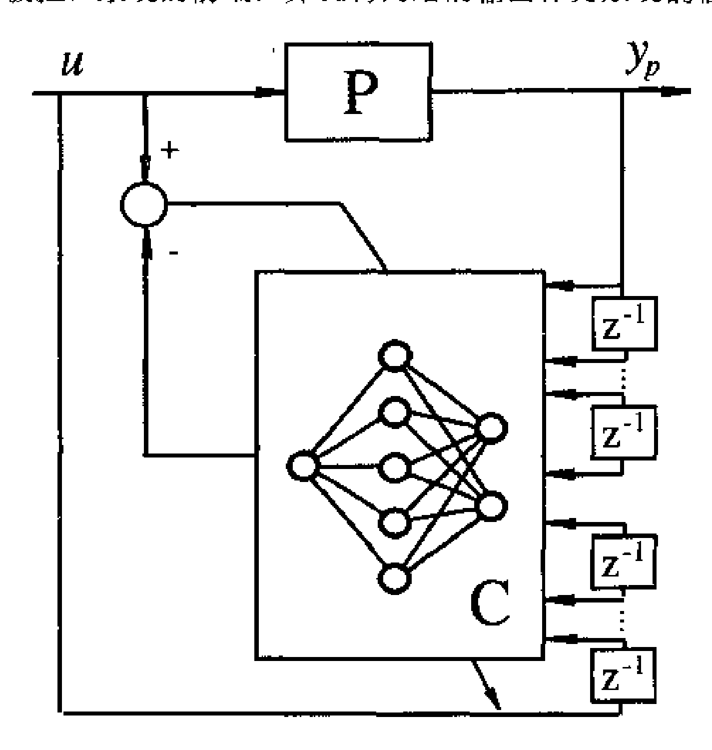


图 3.3 直接逆模型训练图

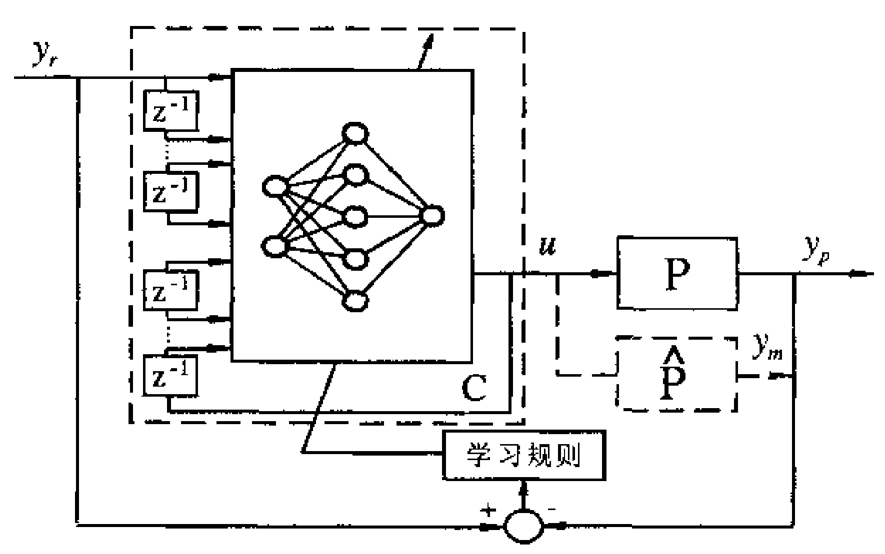


图 3.4 间接逆模型训练图

在实际训练中，也可用一个已训练过的系统模型 $\hat{P}$（如前述的建模网络）替代实际被控系统进行逆模型的训练（如图 3.4 中虚线所示）。采用间接法的特点是：训练过程是

直接指向目标的。因为它是基于期望的系统输出与实际输出之间的误差来调整训练控制器权值的，换句话说，在训练过程中，网络所接受的样本输入对应于它最终工作时接受的实际输入值。另外，在系统的逆关系不是一一对应的情况下，用此法也能对期望的特性找到一个对应的逆。

3.3 系统中的控制

3.3.1 监督式控制

在许多难以用常规控制技术设计自动控制器，而采用手动对特殊任务进行反馈控制的地方，或需要用自动控制器来模仿人的行为的地方，都可以用一个神经网络来实现，网络的训练过程类似于上述的正向模型的训练(见图3.5)，但是，此时的网络结构可以根据需要不再有延时的反馈输出量，只要对应于比如人所感觉的信息，而用于训练网络的目标输出则对应于人对系统的控制量。另外，对于只要能够写出足够范围内一定的对感应到的系统输出误差给出相应控制值的地方，都可以采用这种监督式训练的网络进行非线性自动控制。

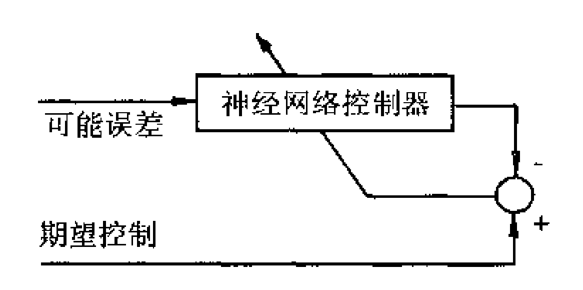


图 3.5　监督式网络控制训练图

3.3.2 直接逆控制

直接逆控制器采用的就是一个逆系统模型，将逆模型网络做为控制器使用，简单地与被控系统相级联，以使复合系统的输出与期望的响应（即网络输入）相一致。很明显，这种控制策略的控制精度极大地依赖于作为控制器的逆模型的精度及其自适应能力。一般情况下可能会产生鲁棒问题。低鲁棒性的主要原因是由于系统为开环控制，没有形成闭环回路，解决这一问题可以通过以下两种方案。

第一种方案是在系统进行控制的同时，并行地对控制器的参数进行在线地修正，如图3.6所示的整个控制系统结构图，其中C2表示控制中使用的逆模型控制器，C1表示在线调整的与C2结构完全相同的逆模型控制器，并定期将产生修正的新权值送入到C2中。

第二种方案是对整个系统加入负反馈回路（见图3.7）。由于逆模型C2的使用，使得前向回路中的非线性得以抵消，所以很容易采用某种简单的常规控制器C1的设计方法，设计一个负反馈回路来满足期望的系统性的要求。

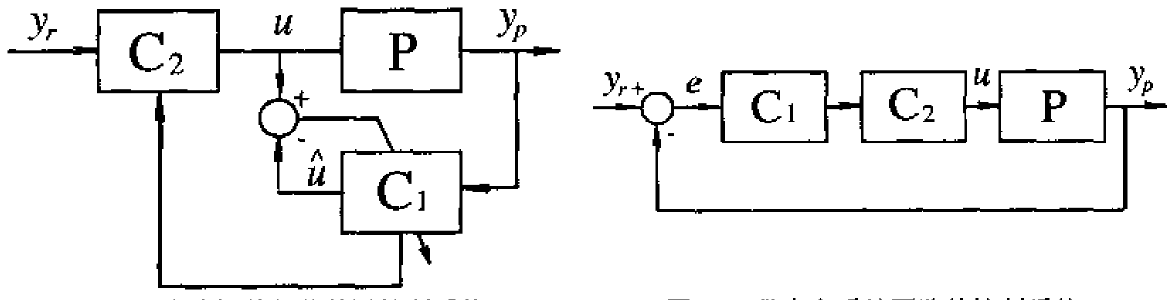


图 3.6 在线调整权值的逆控制系统　　图 3.7 带有负反馈回路的控制系统

3.3.3 模型参考控制

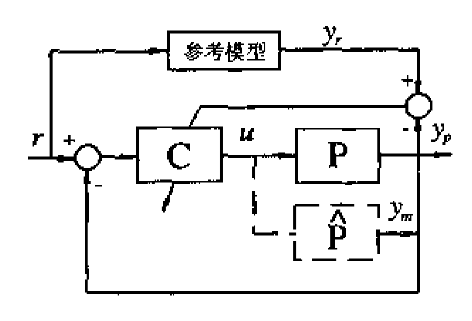


图 3.8 网络模型参考控制系统

自适应控制中的模型参考控制同样可以用于神经网络的控制系统中，此时闭环系统的期望特性由一个适当的参考模型来确定。这个参考模型也可以用一组期望的输入/输出数据对来定义，仔细观察一下可以发现，前面所用过的间接逆模型控制器的训练过程，实际上用了模型参考的概念，不过真正意义下的模型参考控制器的训练是在闭环回路下进行的(图 3.8)。系统控制的目的是使系统输出与参考模型的输出渐近一致，即：

$$\lim_{k \to \infty} \left\| y_r(k) - y_p(k) \right\| \le \varepsilon \tag{3.4}$$

对某一个确定的常数$\varepsilon \ge 0$成立。

类似于逆建模的情况，在训练控制器时，被控系统可以用训练好的神经网络模型来替代。对于控制工程中的模型参考控制，所求的自适应控制律往往会含有较复杂的，甚至难以实现的控制项。通过训练，采用模型参考神经网路控制，可以克服以上的困难。

3.3.4 内模控制

神经网络的正向模型与逆模型在内模控制中得到了充分的应用，内模控制系统的结构图如图 3.9 所示，图中将一个网络正模型与实际系统相并联，并将两者的输出误差作为对输入信号的干扰反馈到网络控制器的输入端。已经证明，对于开环稳定的系统，利用内模控制可以消除干扰和扰动，得到很好的跟踪控制。内模控制已在过程控制中得到广泛的应用。

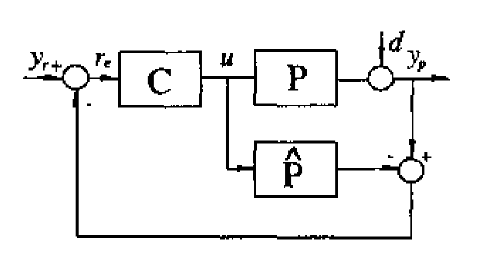


图 3.9 内模控制系统图

第 4 章 反馈网络

反馈网络（Recurrent Network），又称自联想记忆网络，其目的是为了设计一个网络，储存一组平衡点，使得当给网络一组初始值时，网络通过自行运行而最终收敛到所储存的某个平衡点上。

1982 年，美国加州工学院物理学家霍普菲尔德（J. Hopfield）发表了一篇对人工神经网络研究颇有影响的论文。他提出了一种具有相互联接的反馈型人工神经网络模型，并将“能量函数”的概念引入到对称霍普菲尔德网络的研究中，给出了网络的稳定性判据，并用来求解约束优化问题，如 TSP 问题的求解，实现 A/D 转换等。他利用多元霍普菲尔德网络的多吸引子及其吸引域，实现了信息的联想记忆(Associative Memory)功能。另外，霍普菲尔德网络与电子模拟线路之间存在着明显的对应关系，使得该网络易于理解且便于实现。而它所执行的运算在本质上不同于布尔代数运算，对新一代电子神经计算机具有很大的吸引力。

反馈网络能够表现出非线性动力学系统的动态特性。它所具有的主要特性为以下两点，第一，网络系统具有若干个稳定的平衡状态。当网络从某一初始状态开始运动后，网络系统总可以收敛到某一个稳定的平衡状态。第二，系统稳定的平衡状态可以通过设计网络的权值而被存贮到网络中。

如果将反馈网络稳定的平衡状态作为一种记忆，那么当网络由任一初始状态向稳态的转化过程，实质上是一种寻找记忆的过程。网络所具有的稳定的平衡点是实现联想记忆的基础。所以对反馈网络的设计和应用必须建立在对其系统所具有的动力学特性理解的基础上，这其中包括：网络的稳定性、稳定的平衡状态以及判定其稳定的能量函数等基本概念。

针对人工神经网络的特点，若按其运行过程的信息流向来分类，则可分为两大类：前向网络和反馈网络。在前面的章节里，主要介绍了前向网络，通过许多具有简单处理能力的神经元的相互组合作用使整个网络具有复杂的非线性逼近能力。在那里，着重分析的是网络的学习规则和训练过程，并研究如何提高网络整体非线性处理能力。在本章中，我们将集中讨论反馈网络，通过网络神经元状态的变迁而最终稳定于平衡状态,得到联想存储或优化计算的结果。在这里，着重关心的是网络的稳定性问题，研究的重点是怎样得到和利用稳定的反馈网络。

反馈式网络有多种，这里主要讨论由霍普菲尔德提出的反馈网络。霍普菲尔德网络是单层对称全反馈网络，根据其激活函数的选取不同，可分为离散型的霍普菲尔德网络（Discrete Hopfield Neural Network, 简称 DHNN）和连续型的霍普菲尔德网络(Continue Hopfield Neural Network, 简称 CHNN)。DHNN 的激活函数为二值型的，其输入、输出为{-1,1}的反馈网络，主要用于联想记忆，CHNN 的激活函数的输入与输出之间的关系为连续可微的单调上升函数，主要用于优化计算。

霍普菲尔德网络已经成功地应用于多种场合，现在仍常有新的应用的报道。具体的应用方向主要集中在以下方面：图像处理、语音处理、信号处理、数据查询、容错计算、模

式分类、模式识别等。

4.1 霍普菲尔德网络模型

反馈网络的网络结构如图 4.1 所示。

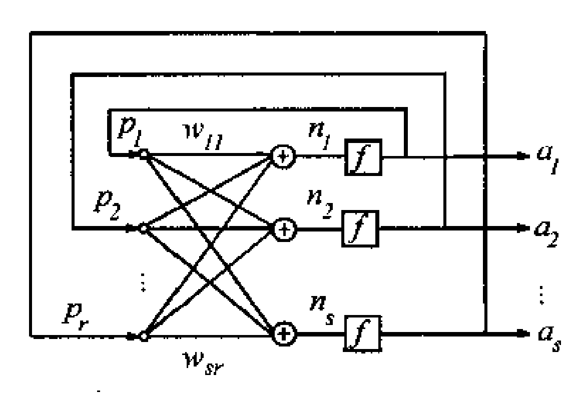


图 4.1 反馈网络结构图

该网络为单层全反馈网络，其中每个神经元的输出都是与其他神经元的输入相连的。所以其输入数目与输出层神经元的数目是相等的，有 $r = s$。

在反馈网络中，如果其激活函数 $f(\cdot)$ 是一个二值型的硬函数，即 $a_i = \text{sgn}(n_i)$，$i = 1, 2, \ldots, r$，则称此网络为离散型反馈网络，如果 $a_i = f(n_i)$ 中的 $f(\cdot)$ 为一个连续单调上升的有界函数，这类网络被称为连续型反馈网络。

4.2 DHNN 的学习规则

4.2.1 海布学习规则

在 DHNN 的网络训练过程中，运用的是海布（Hebb）调节规则：当神经元输入与输出节点的状态相同（即同时兴奋或抑制）时，从第 j 个到第 i 个神经元之间的连接强度则增强，否则则减弱。海布法则是一种无指导的死记式学习算法。

离散型霍普菲尔德网络的学习目的，是对具有 q 个不同的输入样本组 $P_{r\times q} = [P^1 P^2 \ldots P^q]$，希望通过调节计算有限的权值矩阵 W，使得当每一组输入样本 $P^k, k = 1,2,\ldots,q$，作为系统的初始值，经过网络的工作运行后，系统能够收敛到各自输入样本矢量本身。当 $k = 1$ 时，对于第 i 个神经元，由海布学习规则可得网络权值对输入矢量的学习关系式为：

$$w_{ij} = \alpha\, p_j^1 a_i^1 \tag{4.1}$$

其中，$\alpha > 0$，$i = 1, 2, \ldots, r$；$j = 1, 2, \ldots, r$。在实际学习规则的运用中，一般取$\alpha = 1$ 或

$\alpha=\dfrac{1}{r}$。(4.1)式表明了海布调节规则：神经元输入 P 与输出 A 的状态相同（即同时为正或为负）时，从第 j 个到第 i 个神经元之间的连接强度 w_{ij} 则增强(即为正)，否则 w_{ij} 则减弱（为负）。

那么由(4.1)式求出的权值 w_{ij} 是否能够保证 $a_i=p_i=t_i$？我们来验证一下，对于第 i 个输出节点，有：

$$a_i^1=\text{sgn}(\sum_{j=1}^{r}w_{ij}p_j^1)=\text{sgn}(a_i^1)=a_i^1$$

因为 p_i 和 a_i 值均取二值{-1, 1}，所以当其为正值时，输出为 1；其为负值时，输出为 0。同符号值相乘时，输出必为 1。而且由 $\text{sgn}(a_i^1)$可以看出，不一定需要 $\text{sgn}(a_i^1)$的值，只要符号函数 sgn(·) 中的变量符号与 a_i^1 的符号相同，即能保证 sgn(·) = a_i^1。这个符号相同的范围就是一个稳定域。

当 k = 1 时，海布规则能够保证 $a_i=t_i$成立。现在的问题是：对于同一权矢量 W，网络不仅要能够使一组输入状态收敛到其稳态值，而是要能够同时记忆住多个稳态值，即同一个网络权矢量必须能够记忆住多组输入样本，使其同时收敛到不同对应的稳态值。所以，根据海布规则的权值设计方法，当 k 由 1 增加到 2，直至 q 时，则是在原有已设计出的权值的基础上，增加一个新量 $p_j^k a_i^k$，$k=2, ..., q$，所以对网络所有输入样本记忆权值的设计公式为：

$$w_{ij}=\alpha\sum_{k=1}^{q}t_j^k t_i^k \tag{4.2}$$

式中，矢量 T 为记忆样本。上式称为推广的学习调节规则。当系数α = 1 时，称(4.2)式为 T 的外积和公式。

DHNN 的设计目的是使任意输入矢量经过网络循环最终收敛到网络所记忆的某个样本上。因为霍普菲尔德网络有 $w_{ij}=w_{ji}$，所以完整的霍普菲尔德网络权值设计公式应当为：

$$w_{ij}=\alpha\sum_{\substack{k=1\\ i\neq j}}^{q}t_j^k t_i^k \tag{4.3}$$

用向量形式表示为：

$$W=\alpha\sum_{k=1}^{q}[T^k(T^k)^{\mathrm{T}}-I] \tag{4.4}$$

当$\alpha=1$时有：

$$W=\sum_{k=1}^{q}T^k(T^k)^{\mathrm{T}}-qI \tag{4.5}$$

其中，I 为单位对角矩阵。

由(4.4) 式和 (4.5) 式所形成的网络权值矩阵为零对角阵。

采用海布学习规则来设计记忆权值，是因为设计简单，并可以满足 $w_{ij}=w_{ji}$ 的对称条

件，从而可以保证网络在异步工作时收敛。在同步工作时，网络或收敛或出现极限环为 2。在设计网络权值时，与前向网络不同的是令初始权值 $w_{ij}=0$，每当一个样本出现时，都在原权值上加上一个修正量，即 $w_{ij}=w_{ij}+t_j^k t_i^k$，对于第 k 个样本，当第 i 个神经元输出与第 j 个神经元输入同时兴奋或同时抑制时，$t_j^k t_i^k \geq 0$，当 $t_j^k t_i^k$ 中一个兴奋一个抑制时，$t_j^k t_i^k < 0$。这就和海布提出的生物神经细胞之间的作用规律相同。

4.2.2 正交化的权值设计

这一方法的基本思想和出发点是为了满足下面四个要求：

① 保证系统在异步工作时的稳定性，即它的权值是对称的，满足 $w_{ij}=w_{ji}$, $i, j=1, 2, \ldots, s$；

② 保证所有要求记忆的稳定平衡点都能收敛到自己；

③ 使伪稳定点的数目尽可能的少；

④ 使稳定点的吸引力尽可能的大。

正交化权值计算公式推导如下：

a) 已知有 q 个需要存储的稳定平衡点 T^1，T^2，…，T^q，$T \in R^s$，计算 $s\times(q-1)$阶矩阵 $Y\in R^{s\times(q-1)}$：

$$Y=[\,T^1-T^q \quad T^2-T^q \quad \ldots \quad T^{q-1}-T^q\,]^{\mathrm{T}}$$

b) 对 Y 进行奇异质量及酋矩阵分解，如存在两个正交矩阵 U 和 V 以及一个对角值为 Y 的奇异值的对角矩阵 A，满足：

$$Y=UAV$$

$$Y=\left[Y^1\ Y^2\ \ldots\ Y^{q-1}\right]^{\mathrm{T}}$$

$$U=\left[U^1\ U^2\ \ldots\ U^s\right]^{\mathrm{T}}$$

$$V=\left[V^1\ V^2\ \ldots\ V^{q-1}\right]^{\mathrm{T}}$$

$$A=\begin{bmatrix}\lambda_1 & \cdots & 0\\ \vdots & \lambda_k & \vdots\\ 0 & \cdots & 0\end{bmatrix}$$

k 维空间为 s 维空间的子空间，它由 k 个独立基组成：

$$k=\operatorname{rank}(A)$$

设$\{U^1 U^2 \dots U^k\}$为 Y 的正交基，而$\{U^{k+1} U^{k+2} \dots U^s\}$为 s 维空间中的补充正交基。下面利用 U 矩阵来设计权值。

c) 定义：

$$W^+ = \sum_{i=1}^{k} U^i (U^i)^{\mathrm{T}}, \quad W^- = \sum_{i=k+1}^{s} U^i (U^i)^{\mathrm{T}} \tag{4.6}$$

总的联接权值为：

$$W_\tau = W^+ - \tau W^- \tag{4.7}$$

其中，τ为大于-1 的参数。

d) 网络的阀值定义为：

$$B_\tau = T^q - W_\tau T^q \tag{4.8}$$

由此可见，网络的权矩阵是由两部分的权矩阵 W^+和 W^-相加而成的，每一部分权所采用的都是类似于外积型法得到的，只是用的不是原始要求记忆的样本，而是分解后正交矩阵的分量。这两部分权矩阵均满足对称条件，即有下式成立：

$$w_{ij}^+ = w_{ji}^+, \qquad w_{ij}^- = w_{ji}^- \tag{4.9}$$

因而 $W\tau$ 中分量也满足对称条件。这就保证了系统在异步时能够收敛并且不会出现极限环。

下面我们来推导记忆样本能够收敛到自己的有效性。

a) 对于输入样本中的任意目标矢量T^i，$i = 1, 2, \dots, q$，因为$(T^i - T^q)$是 Y 中的一个矢量，它属于 A 的秩所定义的 k 个基空间中的矢量，所以必存在一些系数$\alpha_1, \alpha_2, \dots, \alpha_k$使

$$T^i - T^q = \alpha_1 U^1 + \alpha_2 U^2 + \dots + \alpha_k U^k$$

即

$$T^i = \alpha_1 U^1 + \alpha_2 U^2 + \dots + \alpha_k U^k + T^q$$

对于 U 中任意一个U^i，有

$$\begin{aligned} W_\tau U^i &= W^+ U^i - \tau W^- U^i \\ &= \sum_{i=1}^{k} U^i (U^i)^{\mathrm{T}} U^i - \tau \sum_{i=k+1}^{s} U^i (U^i)^{\mathrm{T}} U^i \\ &= U^i \end{aligned}$$

由正交性质可知，上式中，只有 $i = l$ 那一项与U^l相乘后结果为 1，其余项均有$(U^i)^{\mathrm{T}} U^l = 0$。

对于样本输入T^i，其网络输出为：

$$\begin{aligned}A^i &= \mathrm{sgn}\left(W_\tau T^i + B_\tau\right)\\ &= \mathrm{sgn}\left(W^+T^i - \tau W^-T^i + T^m - W^+T^q + \tau W^-T^q\right)\\ &= \mathrm{sgn}\left[W^+(T^i - T^m) - \tau W^-(T^i - T^q) + T^q\right]\\ &= \mathrm{sgn}\left[(T^i - T^q) + T^q\right]\\ &= T^i\end{aligned}$$

b) 当选择第 q 个样本 T^q 作为输入时，有：

$$\begin{aligned}A^q &= \mathrm{sgn}\left(W_\tau T^q + B_\tau\right)\\ &= \mathrm{sgn}\left(W_\tau T^q + T^q - W_\tau T^q\right)\\ &= \mathrm{sgn}\left(T^q\right)\\ &= T^q\end{aligned}$$

c) 如果输入一个不是记忆样本的 P，则网络输出为：

$$A = \mathrm{sgn}(W_\tau P + B_\tau) = \mathrm{sgn}\left[\left(W^+ - \tau W^-\right)\left(P - T^q\right) + T^q\right]$$

因为 P 不是已学习过的记忆样本，$P - T^q$ 不是 Y 中的矢量，则必然有 $W_\tau(P - T^q) \neq P - T^q$，并且在设计过程中可以通过调节 $W_\tau = W^+ - \tau W^-$ 中的参数τ的大小，来控制$(P - T^q)$与 T^q 的符号，以保证输入矢量 P 与记忆样本之间存在足够的大小余额，从而使 $\mathrm{sgn}(W_\tau P + B_\tau) \neq P$，使 P 不能收敛到自身。

利用参数τ的调节可以改变伪稳定点的数目。在串行工作的情况下，伪稳定点数目的减少就意味着每个期望稳定点的稳定域的扩大。对于任意一个不在记忆中的样本 P，总可以设计一个τ把 P 排除在外。

表 4.1 给出的是一个τ = 10，学习记忆 5 个稳定样本的系统，采用上面的方法进行权的设计，以及在不同的τ时的稳定点数目。同时给出了正交化设计方法与外积和网络权值设计法的比较。

表 4.1 不同的τ时的稳定点数目

	τ = 1	τ = 10	外积型设计
稳定点数	8	5	4
错误稳定点数	0	0	5

虽然正交化设计方法的数学设计较为复杂，但与外积和法相比较，所设计出的平衡稳定点能够保证收敛到自己并且有较大的稳定域。更主要的是在 MATLAB 工具箱中已将此设计方法写进了函数 solvehop.m 中：

[W, b] = solvehop(T);

用目标矢量给出一组目标平衡点，由函数 solvehop.m 可以设计出对应反馈网络的权

值和偏差，保证网络对给定的目标矢量的输入能收敛到稳定的平衡点。但网络可能也包括其他伪平衡点，这些不希望点的数目通过选择τ值（缺省值为 10）已经做了尽可能的限制。

一旦设计好网络，可以用一个或多个输入矢量进行测试。这些输入将趋近目标平衡点，最终找到它们的目标矢量。下面给出用正交化方法设计权值的例子。

例 4.1 考虑一个具有两个神经元的霍普菲尔德网络，每个神经元具有两个权值和一个偏差。

网络所要存贮的目标平衡点为一个列矢量 T：

$$T = [\ 1\ -1;\\ -1\ 1\ ];$$

T 将被用在设计函数 solvehop.m 中以求出网络的权值：

[W, b] = solvehop(T);

solvehop.m 函数返回网络的权值与偏差为：

$$W = [\ 0.6925\ -0.4694;\\ -0.4694\ 0.6925\ ]$$

$$b = 1.0\ e\text{-}16 * [0.6900; 0.6900]$$

可以看出其权值是对称的。

下面用目标矢量作为网络输入来测试其是否已被存贮到网络中了。取输入为：

Ptest = [1 -1; -1 1];　　　　　　　　% 输入矢量

用来进行测试的函数为 simuhop.m：

A = simuhop(Ptest, W, B, 3);　　　　　% 计算其网络循环结束后的输出

函数右端中的第四个参数 3 表示网络反复循环的次数。经过 3 次的运行，如同所希望的那样，网络的结果为目标矢量 T。

现在我们想知道所设计的网络对任意输入矢量的收敛结果。我们首先给出六组输入矢量为：

P = [0.5621 0.3577 0.8694 0.0388 -0.9309 0.0594;
　　-0.9059 0.3586 -0.2330 0.6619 09.8931 0.3423];

在经过 25 次循环运行后，得到输出为：

A = [1.0000　-0.0191　1.0000　-1.0000　-0.8036　-1.0000;
　　-1.0000　0.0191　-1.0000　1.0000　0.8036　1.0000];

如同所看到的第 2 组及第 5 组的输入矢量没有能够收敛到目标平衡点上。然而，如果让其运行 60 次，它们则能够收敛到第 2 个目标矢量上。

另外，当取其他随机初始值时，在 25 次循环后均能够收敛到所设计的平衡点上。一般情况下，此网络对任意初始随机值，在运行 60 次左右均能够收敛到网络所储存的某个平衡点上。

4.3 离散型反馈网络的稳定点与稳定域

反馈网络因结构上输入/输出节点相连而合二为一决定其网络系统的输出状态具有动

态变化特性。当对网络输入任意初始状态，通过反馈回路的不断自动循环，网络的输出状态最终总可以变化运动到某种状态：稳定、循环、发散或混沌。设计者的任务是通过设计反馈网络权值，使其处于有用的稳定状态，即通过网络的自动循环，而最终达到某一个稳定的平衡状态，不再发生变化。如果将期望的稳定平衡点作为一种记忆让网络记住，那么网络由任一初始状态向稳态的转化过程，实质上是一种寻找记忆的过程。状态的初始值则是给定的有关该记忆的部分信息，状态的移动过程则是从部分信息去寻找全局信息的过程。如果把系统的稳定点变成一个能量函数的极小点，则可以把这个能量极小值点作为一个优化目标函数的极小点，把状态变化的过程看成是优化某一个目标函数的过程。以上就是反馈网络的两个最基本功能：联想记忆和优化计算。反馈网络的另一个特点是它的解并不需要真正去计算，只需要适当选择反馈网络结构，按某一种方法设计其权值，然后使其自行工作运行，即可使初始输入在状态的不断变化过程中，自动收敛到所属的稳定状态。

在反馈网络的研究中，研究最多的还是网络权值的设计方法。因为要想利用从初始状态到稳定点的运行过程来实现对信息的联想存取，对反馈网络权值的设计必须达到下列目标：第一，网络系统能够达到稳定的收敛，即研究系统的稳定条件。第二，网络的稳定点使网络能够收敛到所设计的稳定平衡点上。第三，希望所记忆的稳定点有尽可能大的吸引域，而非希望的稳定点的吸引域尽可能的小。

根据输入/输出节点的兴奋或抑制状态一致性原理进行权值设计的海布学习规则的优点是设计简单，但由于它所采用的方法是将所要记忆的样本，全部累加到权值上的方法，使权值随着所需记忆的样本数量的增加，而不断产生移动变化，具有记忆样本之间的交叉干扰，产生遗忘和疲劳现象，对于样本的记忆产生不可靠的结果。正交化权值设计方法能够较好解决此问题。它能够保证所要求记忆的稳定平衡点都能收敛到自己，并使伪稳定点(即未要求网络记忆，但是网络的稳定平衡点)尽可能的少。虽然此方法的设计稍微复杂了点，不过在有关软件工具，如 MATLAB 环境下的神经网络工具箱的帮助下，可以很方便地设计出网络来。所以只需设计者从理论上、概念上掌握要领，可把精力投入到具体应用上。

本节的重点在于对稳定点及其稳定域进行分析，主要是想回答下列问题：①当反馈网络的结构确定以后，网络的稳定平衡点的数目是否确定？②哪些输入是稳定的平衡点？③随着所要求记忆的稳定平衡点数目的增加，其吸引域如何变化？其缩小的规律如何？④不稳定的平衡点有哪些？

本节是通过实际设计网络记忆平衡点，然后使其运行工作，并采用简单明了的作图方式，直观地观察在不同网络结构情况下的平衡点收敛趋势及收敛域的范围，从而揭示反馈网络在不同网络结构下的轨迹变化规律，及其进行联想记忆、容错和优化计算的实质，使人们对反馈网络的工作原理及其应用有更深入的认识。下面首先从简单的两个输入神经元节点的网络结构入手，详细探讨每一种记忆的可能性，并总结出共性。然后对复杂的三个输入神经元节点的网络情况进行研究，并对四个输入神经元节点的网络情况进行推论归纳。最后对在不同结构情况下网络平衡点的收敛趋势及收敛域的范围给予小结。

4.3.1 两个输入神经元的情况

反馈网络权值的设计是采用正交化设计方法。与一般定义的离散霍普菲尔德网络不同，用正交化设计方法进行权值设计得到的反馈网络的权值 $w_{ii} \neq 0$，且偏差 $B \neq 0$。不过满足网络的单层对称性，即有 $w_{ij} = w_{ji}$。为了能够清晰地看到网络输出变化的轨迹，在实验的网络运行工作的过程中，给网络采用了如图 4.2 所示的一个输出范围为[-1, 1]的饱和函数。这使得网络在运行的过程中，也就是网络状态达到其稳态-1 或 1 的过程中，可以输出一个从初态逐渐变化到稳态的轨迹，使我们可以很清楚地看到网络运行过程中的暂态变化的过渡过程，在一定程度上为我们的认识网络的特性提供了方便。

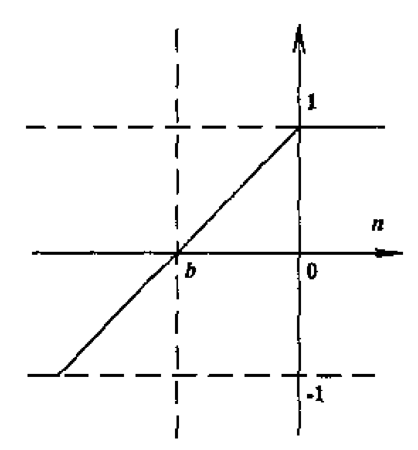


图 4.2 反馈网络的激活函数

由离散型反馈网络激活函数可以知道，对于具有两个输入神经元的反馈网络，系统的可能平衡点应当为：

$$(1)\begin{bmatrix}1\\1\end{bmatrix};\ (2)\begin{bmatrix}1\\-1\end{bmatrix};\ (3)\begin{bmatrix}-1\\1\end{bmatrix};\ (4)\begin{bmatrix}-1\\-1\end{bmatrix}$$

根据前面 4.2 中的分析可知，不论采用何种方法，当网络只记忆一个平衡点时，定能保证网络对该稳定点输入准确地收敛到自身，并且对网络所有其他任意的输入点均能收敛到该平衡点。而当要网络记忆较多平衡点时，就有可能出错。正交化的权值设计方法能够保证使需要记忆的稳定平衡点都能收敛到自身。首先来观察一下记忆四个平衡点中的两个平衡点时的稳定域情况。

1） 设计网络记忆的稳定平衡点为

```
T = [1  -1;
    -1   1]
```

利用 MATLAB 环境下的神经网络工具箱来进行网络的设计十分简单，只要根据期望网络记忆的平衡点 T 的值，通过调用采用正交法权值的设计函数[W, B] = solvehop(T)即可完成。网络设计完成后可得权值为

```
W =[ 0.6925 -0.4694
```

-0.4694 0.6925];

偏差为

B = 1.0e-016 *[0.6900
0.6900];

从网络的权值上可以看出，所设计的网络权值是对称的，即有 $w_{ij}=w_{ji}$，并且 $w_{ii}\neq 0$，偏差 $B\neq 0$。

为了观察网络工作效果，向设计好的网络输入任意测试数据

P=[0.5000 0.9003 0.7826 -0.0871 -0.8000 -0.6012 0.1200 -0.2145
0.5000 -0.5377 0.7242 -0.9630 -0.8000 0.2317 0.7068 -0.0890];

对于两个输入节点的反馈网络的输出变化情况，可以用分别表示两个节点分量的平面坐标来画出网络输出的运动变化轨迹，如图 4.3 所示，图中的横坐标为第一个神经元节点的输出，纵坐标为第二个神经元节点的输出，图中"+"表示被网络记忆的稳定平衡点，"×"为网络初始输入数据点，"o" 表示网络在连续不断循环工作下的输出轨迹。对于所设计的反馈网络，[1; -1]和[-1; 1]点为稳定平衡点，由图 4.3 可以看出该网络将整个坐标平面以两个平衡点连线的垂直平分线为分界线平分为两部分，并以平衡点之间连线作为收敛路径走向，所以每一个平衡点的稳定域是相当大的:各占半个平面。那么平衡点连线间的垂直平分线，包括另外两个未被记忆的平衡点[-1; -1]和[1; 1]的运动趋势是什么呢？从图中曲线走向可以看到，它们收敛到了原点[0; 0]。原点[0; 0]是个什么类型的点呢？只要我们将网络的运行循环次数加大，便可发现，经过 478 次循环后，[-1; -1]和[1; 1]两点连线上的任何点，在收敛到原点[0; 0]之后，也转向收敛到[1; -1]或[-1; 1]点了。所以，原点[0; 0]是一个不稳定的平衡点。这主要是由计算机的计算精度造成的一定的计算误差，即使这个偏差非常小，但一旦偏离了[-1; -1]和[1; 1]两点的连线，则落到了平衡稳定点的区域内而转向收敛到稳定的平衡点。

2）设计网络记忆平衡点为:

T = [1 -1;
1 -1]

以同样的方式设计网络权值，可得权值:

W =[0.6925 0.4694
0.4694 0.6925];

偏差为:

B =1.0e-016 *[0.6900
-0.6900]。

向网络输入任意测试矢量:

P = [-0.8000 0.5000 -0.1098 -0.0680 0.6924 -0.5947 0.6762 0.3626

0.8000 -0.5000 0.8636 -0.1627 0.0503 0.3443 -0.9607 -0.2410];

所设计的网络对给定的测试矢量 P 的输出轨迹图如图 4.4 所示。从中可以看出，与[1;

-1]或[-1; 1]为平衡点的第一种情况类似，以[1; 1]和[-1; -1]为平衡点的反馈网络的稳定域也将整个平面划分为两个区域，并以两点的垂直平分线为界，同样存在该垂直平分线上的所有点都收敛于不稳定的平衡点[0; 0]的情况。

3) 改变记忆对角线上的点的做法，将表示网络输出的平面内某一个边线上的两个含有1的点作为平衡点让网络记忆，即设计网络记忆平衡点

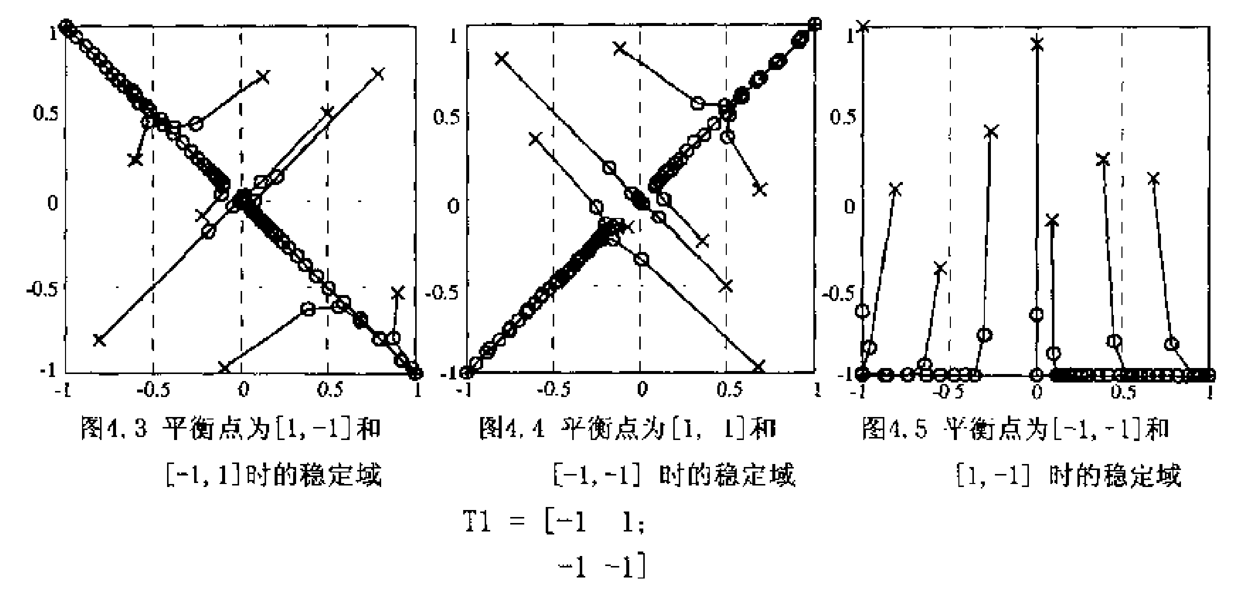


图4.3 平衡点为[1,-1]和[-1,1]时的稳定域　　图4.4 平衡点为[1, 1]和[-1,-1] 时的稳定域　　图4.5 平衡点为[-1,-1]和[1,-1] 时的稳定域

T1 = [-1 1;
-1 -1]

再次设计网络权值，可得权值：

W =[1.1618 0
0 0.2231]

偏差为：

B =[0
-0.8546]

同样，对所设计的网络输入一组测试数据：

P = [0 0 -0.3176 0.4542 0.6770 -0.2592 0.0931 0.3891
0.8000 -0.5000 0.0682 -0.3814 0.1361 0.4055 -0.1102 0.2426];

可得如图 4.5 所示网络运行轨迹图。与前两种收敛域有所不同，此时的收敛域分为左右两个平面，它们以[-1; -1]和[1; -1]两点连线的垂直平分线为分界线。垂直平分线上的所有点都收敛到[0; -1]点。同样，这个点也为不稳定的平衡点。同理可知，当选择

$$T2=\begin{bmatrix}-1 & 1\\ 1 & 1\end{bmatrix};\quad T3=\begin{bmatrix}-1 & -1\\ 1 & -1\end{bmatrix};\quad T4=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

分别作为网络的收敛平衡点让其记忆时，其收敛域分别以平衡点连线的垂直平分线为界线的两部分，并且垂直平分线与两点连线的交点为不稳定的平衡点。

3) 取四个平衡点中的三个点作为平衡点记忆

T = [-1 -1 1;
-1 1 -1]

以同样的方法可以设计出网络权值为：

W =[0.6925 -0.4694

-0.4694 0.6925]

偏差为：

B = 1.0e-016 * [0.6900
0.6900]

向网络输入任意测试数据：

P = [-0.6122 0.1384 -0.5312 0.8632 0.3111 0.2546 -0.2056 0.3104
0.8096 0.2636 0.0976 -0.3296 -0.2162 0.3982 -0.1727 0.6752]

这里首先值得一提的是，此时所获得的网络权值与网络记忆两个平衡点 1)时的情况的权值完全相同。但将两图(图 4.3 与图 4.6(a))对比可见，网络的输出轨迹是大不相同的。这说明，网络对不同平衡点的权值设计过程是具有动态记忆的。由于增加了所要记忆的平衡点数目，所记忆的稳定点的稳定域减小了，由两个稳定点时的半个平面减少到三个稳定点时的四分之一平面。以[0; 0]作为坐标轴的原点，记忆三个稳定点时的稳定域正好是每个稳定点所处的象限。四个稳定点中没有被记忆的点[1; 1]所在的第一象限中的网络输入将全部收敛到[1; 1]上，该点为伪稳定点。从图 4.6(a)中可以清楚地看出，在输入的测试点 P 中处于第一象限的三个点[0.1384; 0.2636]，[0.2546; 0.3982]，[0.3104; 0.6752]，都收敛到未被设计为网络记忆的平衡点[1; 1]上了。该结论可以推广到其他三种情况，即对于只记忆四个点中三个稳定点的网络，未被记忆的点在网络工作时也有一个收敛域，并可能作为伪稳定点出现。作为四个收敛域的的分界线——坐标轴上的点将分别收敛到 [-1; 0]，[1; 0]，[0; -1]，[0; 1]或[0; 0] 上，它们是不稳定的平衡点。

图 4.6(b)给出平衡点为[-1; 1]，[1; -1]和[-1; -1]时，网络经过 241 次的运行工作，将不稳定的平衡点收敛到各自平衡点上的轨迹图。

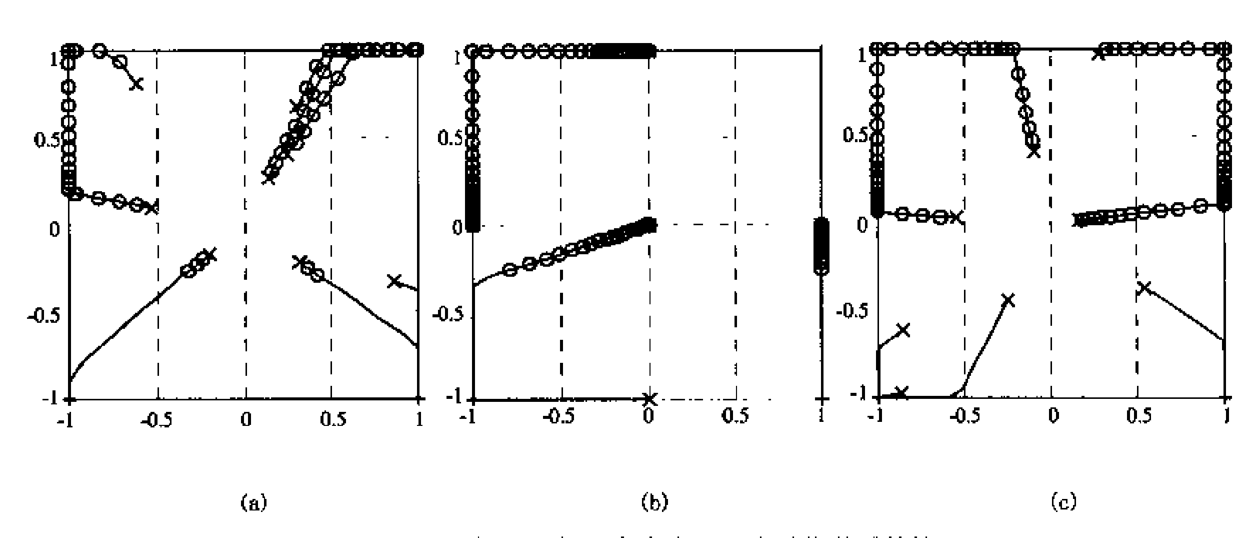


图 4.6 网络记忆的平衡点多于两个时收敛域的情况

4) 设计网络将四个点全部作为平衡点记忆

T = [-1 -1   1 1
-1   1 -1 1]

由此得网络权值为：

W = [1.1618 -0.0000

-0.0000　1.1618]

偏差为：

B = 1.0e-016 *[0.5390
0.1797]

向网络输入任意测试数据

P = [-0.8587 -0.5457 -0.0836 0.1650 -0.8514 -0.2408 0.5418 0.2764
-0.9761 0.0325 0.4064 0.0184 -0.6135 -0.4471 -0.3721 0.9731]

可得如图 4.6(c)所示的运行轨迹图。

根据前面情况的结果已经很容易推导出此时的收敛域为各平衡点所在的象限，并且与情况 3)相同，即点[-1; 0]，[0; 1]，[0; 0]，[1; 0]和[0; -1]均为不稳定的平衡点。不过实验表明，它们在网络运行循环次数较多的情况下，也均会收敛到某一个平衡点上，这主要是由于计算机的计算误差（即使这个误差只有 10^{-14} 数量级）使网络输出值偏离了不稳定点的收敛域，落到了稳定点的收敛域中造成的，这一结果的出现正好符合不稳定平衡点的特性。

4.3.2　网络输入神经元为三个时的情况分析

当网络输入神经元为三个时，其输入/输出变量也由两个变成了三个，从而使得网络的平衡点增至为 2^3 =8 个，描述网络输出运行轨迹的图形也由平面变成了立体空间。根据上一节对两个输入节点的情况与分析，可以比较容易地推导出以下的结论。

①当网络只记忆两个平衡点时，其收敛域应为由两点连线的垂直平分面所截成的两点所在的空间，垂直平分面上的点均收敛到对应的含有 0 的坐标点上，这些坐标点均是不稳定的平衡点。

②当网络记忆三个平衡点时，其收敛域为由三个点所组成的三角形平面上，从每条边上垂直截下的平面所组成的三个子空间。同样，三个垂直平面上的点均收敛到某个含 0 的坐标点上，这些坐标点均为不稳定的平衡点。

③当网络记忆四个以及多于四个平衡点时，其收敛域为空间上的八个象限，且有未被记忆的平衡点，在网络运行工作时，可能会以伪稳定点的形式出现。正交化设计方法的另一个优点则是使网络工作时，尽量少地出现伪稳定点。

下面做出当网络记忆全部八个平衡点时的图形，此时被记忆的平衡点为：

T = [-1 -1 -1 -1 1 1 1 1
-1 1 -1 1 -1 1 -1 1
-1 -1 1 1 -1 -1 1 1]

设计出的网络权值为：

W = [1.1618 -0.0000 0.0000
-0.0000 1.1618 -0.0000
0.0000 -0.0000 1.1618]

偏差为：

B = 1.0e-016 * [0.0000
-0.3593
0.3593]

向网络输入任意一组测试数据：

P = [-0.3908 0.3644 -0.6983 0.7200 -0.0069 0.2898 -0.3161 0.0682
-0.6207 -0.3945 0.3958 0.7073 0.7995 0.6359 -0.4205 0.4542
-0.6131 0.0833 -0.2433 0.1871 0.6433 0.3205 -0.3176 -0.3814]

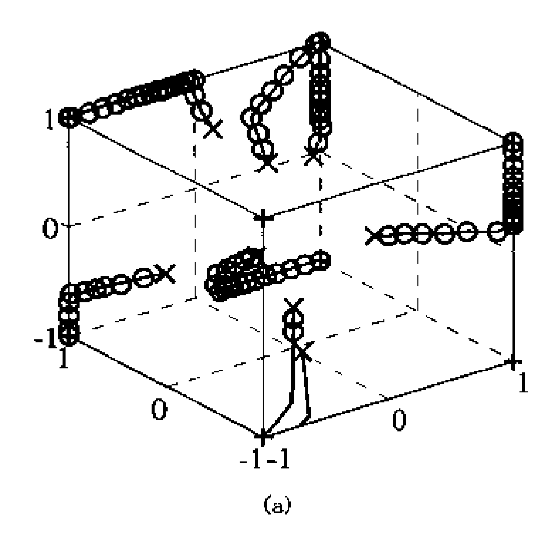
网络在初始输入值下运行到各自平衡点的轨迹图如图 4.7(a)所示。从中可以明显地看到各个点都在自己所处的象限内收敛到所在象限的稳定平衡点上。不过，网络还存在不稳定的平衡点，例如向设计好的网络输入以下点：

[0.8 0.8 -0.8 -0.8 0.3 0.6 -0.7 -0.3 0 0 0 0
0.8 -0.8 0.8 -0.5 0 0 0 0 0.2 0.7 -0.7 -0.4
0 0 0 0 0.7 -0.2 0.6 -0.5 0.4 -0.4 0.8 -0.3]

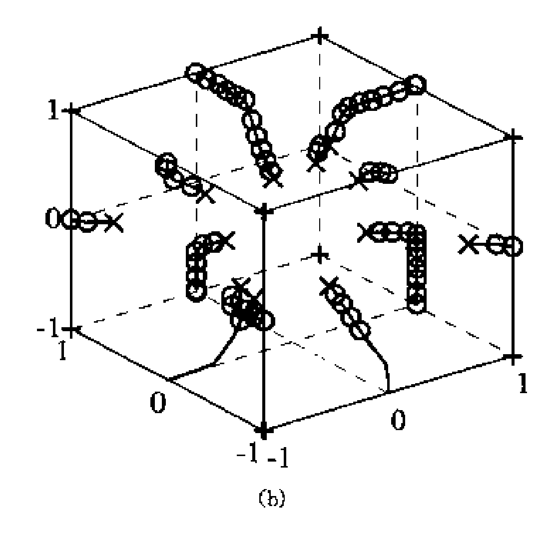
所得的输出运行轨迹如图 4.7(b)所示，可见这些点分别收敛到了以下各点：

[1 1 -1 -1 1 1 -1 -1 0 0 0 0
1 -1 1 -1 0 0 0 0 1 1 -1 -1
0 0 0 0 1 -1 1 -1 1 -1 1 -1]

这些点均为不稳定的平衡点，因为当增大网络的循环次数后，所有上述 12 个点均分别收敛到 8 个稳定的平衡点上。如在同样的输入下，使网络达到循环 254 次时，则输出变为：



(a)



(b)

图 4.7 神经元节点为三个时的稳定点与稳定域的情况

[1 1 -1 -1 1 1 -1 -1 -1 -1 1 1
1 -1 1 -1 -1 -1 -1 -1 1 1 -1 -1
1 1 1 1 1 -1 1 -1 1 -1 1 -1]

实际上，只要包含 0 的坐标点都是不稳定的平衡点，所以四个神经元节点的网络除了上述 12 个不稳定的平衡点外，还有另外 7 个不稳定的平衡点，它们分别是[0; 0; 0]，[0;

0; 1], [0; 0; -1], [0; 1; 0], [0; -1; 0], [1; 0,0], [-1; 0; 0]。

就不稳定的平衡点小结如下：对于具有二神经元节点的网络，最多可能的不稳定的平衡点有 5 个，也就是平分线上的点；对于具有三神经元节点的网络，有 19 个可能的不稳定平衡点；由此可推出对于具有四神经元节点输入时，最多可能的不稳定的平衡点有：

$$C_4^1 * 2^3 + C_4^2 * 2^2 + C_4^3 * 2^1 + C_4^4 * 2^0 = 65\ (\text{个})$$

即坐标中所有含有 0 的点。

通过以上的分析，我们已经可以借助于想象来推出四个神经元节点以上的网络输出运行轨迹的情况：每个平衡点都具有自己的一个稳定域，且其大小取决于网络记忆平衡点数目的多少。当网络记忆全部可能的平衡点时，稳定域为最小。不过当少于最多的平衡点时，网络运行时可能会出现收敛到伪稳定点的情况。这点在实际应用时应当特别注意，尽量避免这种情况的出现，它可能会出现在进行模式辨识时模式缺损太多的情况下。另外一个结论是：在设计网络作为模式辨识时应当尽量多些利用可能有的稳定的平衡点，以减少不稳定的平衡点的数量。这与人们直觉上认为应使网络尽量少记忆平衡点以扩大稳定域的结论是不同的。因为各个稳定点的稳定域的大小是能够得到保证的。但伪稳定点的出现可能使得网络得出错误的决定或未定义的结果。

4.3.3 小结

通过以上实验可以得出结论：对于具有 r 个输入神经元、输出函数为±1的反馈网络，当采用正交权值设计法，所设计的网络最多具有 3^r 个平衡点，其中，2^r 个是稳定的平衡点，另外 3^r-2^r 个是不稳定的平衡点。每个稳定点的稳定域（吸引域）为以该点为中心，棱长为 2 的多边体（形），各平衡域的边界部分为不稳定平衡点的吸引域，不稳定的平衡点在两两稳定平衡点连线的中点上。

4.4 连续型霍普菲尔德网络

霍普菲尔德网络可以推广到输入和输出都取连续数值的情形。这时网络的基本结构不变，状态输出方程形式上也相同。若定义网络中第 i 个神经元的输入总和为 n_i，输出状态为 a_i，则网络的状态转移方程可写为：

$$a_i = f\left(\sum_{j=1}^{r} w_{ij} p_j + b_i\right) \tag{4.10}$$

其中，神经元的激活函数 f 为 S 型的函数（或线性饱和函数）：

$$f1 = \frac{1}{1+\mathrm{e}^{-\lambda(n_i)}} \tag{4.11}$$

或

$$f2 = \tanh(\lambda(n_i)) \qquad (4.12)$$

两个函数共同的特点是当 $n_i \to \infty$ 及 $n_i \to -\infty$ 时，函数值饱和于两极，从而限制了神经网络中输出状态 a_i 的增长范围。显然，若使用函数 $f1(\cdot)$，则 $a_i \in [0,1]$，若使用 $f2(\cdot)$，则 $a_i \in [-1,1]$。函数 $f1(\cdot)$ 和 $f2(\cdot)$ 中的参数 λ 用以控制 S 型函数在 0 点附近的变化数。

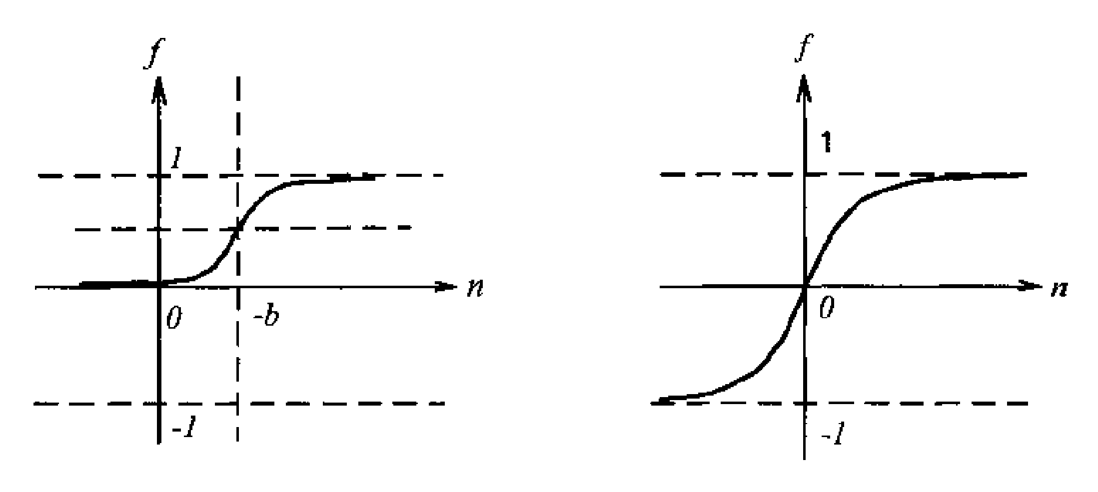


图 4.8 连续霍普菲尔德网络激活函数

4.4.1 对应于电子电路的网络结构

电子电路与 CHNN 之间存在着直接的对应关系。第 i 个输出神经元的模型如图 4.9 所示，其中，运算放大器模拟神经元的激活函数，所以电压 u_i 为激活函数的输入，也称为网络的状态，各并联的电阻 R_{ij} 值决定各神经元之间的连接强度；电容 C 和电阻 r 模拟生物神经元的输出时间常数，另外电流 I_i 模拟阀值。其中，$i, j = 1, 2, ..., r$。整个网络是由 r

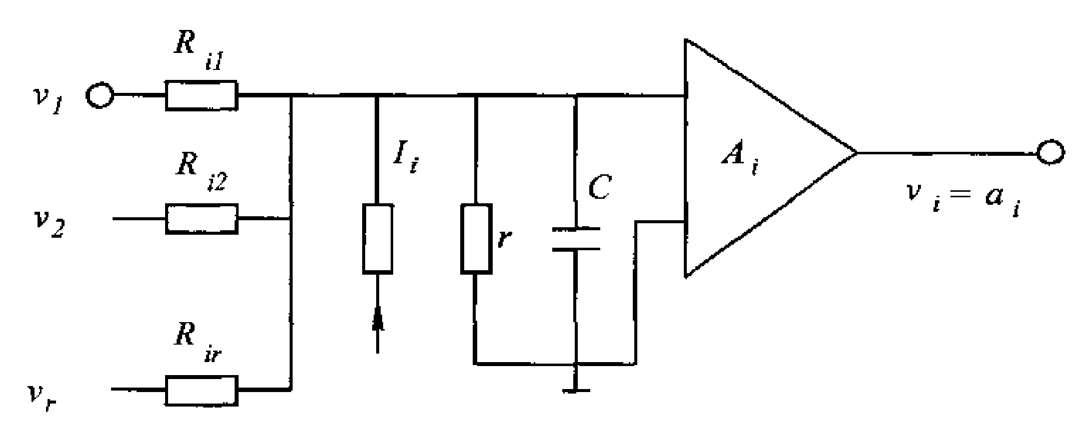


图 4.9 第 i 个输出神经元电路模型

个同样的模型并联组成，每个模型都具有相同的输入矢量 $V = [v_1, v_2, ..., v_r]$，状态矢量和 u_i 由每个模型的输出 v_i 组合而成。电路状态与输出之间的关系图如图 4.10 所示。这也是放大器的特性图。

对于图 4.9 所示的电路图，根据克西霍夫电流定律，有下列方程成立：

$$C\frac{\mathrm{d}u_i}{\mathrm{d}t}+\frac{u_i}{r}=\sum_{j=1}^{r}\frac{1}{R_{ij}}(v_j-u_i)+I_i$$

对上式进行移项及合并，可得：

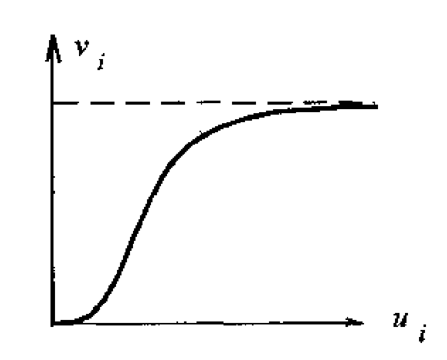


图 4.10 网络输出 v_i 与状态 u_i 的关系图

$$C\frac{\mathrm{d}u_i}{\mathrm{d}t}=-(\frac{u_i}{r}+\sum_{j=1}^{r}\frac{1}{R_{ij}})u_i+\sum_{j=1}^{r}\frac{1}{R_{ij}}v_j+I_i$$

若令

$$\frac{1}{R}=\frac{1}{r}+\sum_{j=1}^{r}\frac{1}{R_{ij}},\qquad w_{ij}=\frac{1}{R_{ij}}$$

则可得：

$$C\frac{\mathrm{d}u_i}{\mathrm{d}t}=-\frac{1}{R}u_i+\sum_{j=1}^{r}w_{ij}v_i+I_i$$

由此可得电路输入的累加值为：

$$n_i=\sum_{j=1}^{r}w_{ij}v_j+I_i \tag{4.13}$$

此式与人工神经网络模型的加权值一致。

电路状态值与加权输入值之间的关系可用一阶微分方程式表示：

$$C\frac{\mathrm{d}u_i}{\mathrm{d}t}=\frac{1}{R}u_i+n_i \tag{4.14}$$

而电路输出与状态之间的关系为一个单调上升的有界函数：

$$v_i=f(u_i) \tag{4.15}$$

方程(4.14)反映了网络状态连续更新的意义，它是与离散形式不同之处，（4.13）～（4.15）式结合在一起描述了 CHNN 的动态过程随着时间的流逝，网络趋于稳定状态，可以在输出端得到稳定的输出矢量。对于由 m 个相互连接的电路模型组成的网络，每个

模型均满足（4.13）~（4.15）方程，可将其写成矩阵形式，为了简便起见，令 $C=1$。

$$\begin{bmatrix} \dot{u}_1(t) \\ \dot{u}_2(t) \\ \vdots \\ \dot{u}_m(t) \end{bmatrix} = -\frac{1}{R}\begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_{m1}(t) \end{bmatrix} + \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mm} \end{bmatrix}\begin{bmatrix} v_1(t) \\ v_2(t) \\ \vdots \\ v_{m1}(t) \end{bmatrix} + \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_m \end{bmatrix}$$

或

$$\dot{U} = -\frac{1}{R}U + WV + I \tag{4.16}$$

这是一个 r 维的线性微分方程。当 $\dot{u}_i(t)=0, i=1,2,\ldots,r$ 时，上式变为：

$$U = R(WV+I)$$

若把 $I=[I_1\ I_2\ \ldots\ I_r]^{\mathrm{T}}$ 看作阀值，那么上式就与人工神经网络的加权输入和 N 形式相似。如果此时代表激活函数 $F(U)$ 的运算放大器的放大倍数足够大到可视为二值型的硬函数，连续型反馈网络即变为离散型的反馈网络。所以也可以说，DHNN 是 CHNN 的一个特例。

4.4.2 霍普菲尔德能量函数及其稳定性分析

霍普菲尔德在 80 年代初提出了一个对单层反馈动态网络的稳定性判别的函数，这个函数有明确的物理意义，是建立在能量基础上的，同李雅普诺夫函数一样，霍普菲尔德认为在系统的运动过程中，其内部贮存的能量随着时间的增加而逐渐减少，当运动到平衡状态时，系统的能量耗尽或变得最小，那么系统自然将在此平衡状态处渐近稳定，即有 $\lim\limits_{t\to\infty}U(t)=U_{\mathrm{e}}$。因此，若能找到一个可以完全描述上述过程的所谓能量函数，那么系统的稳定性问题就可解决。为此对霍普菲尔德反馈网络定义了一种能量函数 E，称为霍普菲尔德能量函数，这个 E 可正可负。但负向有界，所以说，它是李雅普诺夫函数的一种推广，是广义的李雅普诺夫函数。对于连续反馈网络的电路实现，其状态方程组为：

$$\begin{cases} C\dfrac{\mathrm{d}u_i}{\mathrm{d}t} = -\dfrac{u_i}{R} + \sum\limits_{j=1}^{r} w_{ij}v_j + I_i \\ v_i = f_i(u_i) \end{cases} \tag{4.17}$$

当系统达到稳定输出时，霍普菲尔德能量函数定义为：

$$E = -\frac{1}{2}\sum_{i=1}^{r}\sum_{j=1}^{r} w_{ij}v_iv_j - \sum_{i=1}^{r} v_iI_i + \sum_{i=1}^{r}\frac{1}{R}\int_0^{v_i} F^{-1}(\eta)\mathrm{d}\eta \tag{4.18}$$

其中：$\dfrac{1}{R}=\dfrac{1}{r}+\sum\limits_{j=1}^{r} w_{ij}$，$w_{ij}$ 为第 j 个输入与第 i 个输入之间的连接导纳，$w_{ij}=\dfrac{1}{R_{ij}}$；r 与 C 分别为第 i 个运算放大器的电阻和输入电容。I_i 为外加电流，u_i 和 v_i 分别为第 i 个运算放大器的输入与输出。它们之间的关系为一个单调上升的函数关系，如图 4.11(a)所示，其

中β表示运算放大器的放大倍数，图中给出了不同β下输入与输出之间的关系。

在能量函数(4.18)式中，函数 $F^{-1}(v_i)$ 为 u_i 的逆函数。u_i 为 v_i 的函数关系图如图 4.11(b) 所示：能量函数中的积分项表示了输入状态与输出之间关系的能量项，其积分结果如图 4.11(c) 所示。

对于理想放大器，(4.18) 式所定义的能量函数可以被简化为：

$$E = -\frac{1}{2}\sum_{i=1}^{r}\sum_{j=1}^{r} w_{ij} v_i v_j - \sum_{i=1}^{r} v_i I_i$$

这也就成了离散网络的霍普菲尔德能量函数。

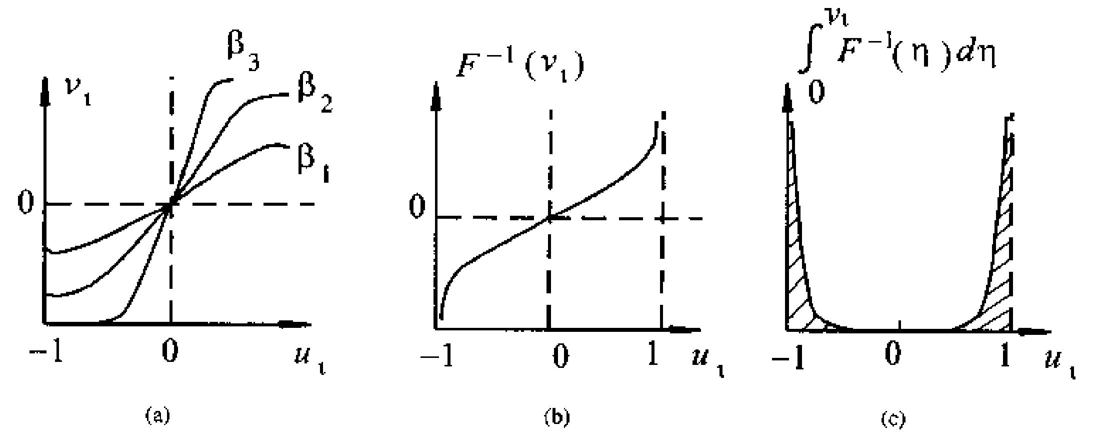


图 4.11 能量函数中各函数之间的函数关系式

对于所定义的霍普菲尔德能量函数 (4.18) 式有以下结论：对于(4.17)式定义的 CHNN 模型系统，若函数 $v_i = f(u_i)$ 是单调递增且连续可微，则能量函数 (4.18) 式是单调递减且有界。

下面首先分析一下能量函数的单调递减。已知：

a) $w_{ij} = w_{ji}$。

b) $f(u_i)$ 为单调递增连续函数。

$$\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}t} &= \sum_{i=1}^{m} \frac{\mathrm{d}E}{\mathrm{d}v_i} \cdot \frac{\mathrm{d}v_i}{\mathrm{d}t} \\
&= \sum_i \frac{\mathrm{d}v_i}{\mathrm{d}t}\left[-\sum_j w_{ij} v_j - I_i + \frac{1}{R} \cdot f^{-1}(v_1)\right] \\
&= -\sum_i \frac{\mathrm{d}v_i}{\mathrm{d}t}\left[\sum_j w_{ij} v_j - I_i - \frac{u_i}{R}\right] \\
&= -\sum_i C \cdot \frac{\mathrm{d}v_i}{\mathrm{d}t} \cdot \frac{\mathrm{d}u_i}{\mathrm{d}t} \\
&= -\sum_i C \cdot \frac{\mathrm{d}v_i}{u_i} \cdot \left(\frac{\mathrm{d}u_i}{dt}\right)^2 \\
&= -\sum_i C \cdot \dot{f}(u_i) \cdot \left(\frac{\mathrm{d}u_i}{\mathrm{d}t}\right)^2
\end{aligned}$$

由已知条件 b)可得 $\dot{f}(u_i)\geq 0$，又因为 $C>0$，所以有 $\frac{\mathrm{d}E}{\mathrm{d}t}\leq 0$。

当 $\frac{\mathrm{d}E}{\mathrm{d}t}=0$ 时，有 $\frac{\mathrm{d}u_i}{\mathrm{d}t}=0, i=1,2,\ldots,r$，这表明，能量 E 的极小点与 $\frac{\mathrm{d}u_i}{\mathrm{d}t}=0$ 的平衡点是一致的。

下面讨论 E 的有界性。这里最主要的是 E 不能下降到负无穷大，E 在负方向要有界。只有这样，当 E 随着时间的增加必定会达到一个极限值，而此极限值正是系统的稳定点。

①因为对于状态和输出，有 $|v_i|\leq 1$，w_{ij} 是由电子元器件的有界数组成的。所以 E 中第一项是有界的。

②因为外加电流 I_i 也是有限值。所以 E 中第二项是有界的。

③对于 E 中第三项，式中 $F^{-1}(\cdot)$ 反映了神经网络的状态与输出之间的关系，也是运算放大器的输入与输出之间的函数关系。若用β代表运算放大器的增益，那么 u_i 与 v_i 的关系可用 $v_i=f(u_i,\beta)$ 来代替 $v_i=f(u_i)$，并由此可得 $u_i=\frac{1}{\beta}F^{-1}(v_i)$。

由此可得第三项为 $\frac{1}{\beta}\sum_{i=1}^{r}\frac{1}{r}\int_0^{v_i}F^{-1}(\eta)\mathrm{d}\eta$，从图 4.11(c)中可以看出，上式的积分对一个 i 来说在 $v_i=0$ 时为零，而在其他情况下为正，当 $\beta\to\infty$ 时，$v_i(u_i)$ 趋向一个符号函数，此时，积分项的作用很小而可以忽略不计，能量函数就由第一、二两项的和决定，成为离散时的情况，E 有界。如果β比较小，第三项为正，它的贡献主要在靠近 $v_i\approx\pm 1$ 的边缘，其值总是小于 $2|v_i||u_i|$，所以也是有界的。

当我们对反馈网络应用霍普菲尔德能量函数后，从任意一个初始状态开始，因在每次迭代后都能满足 $\Delta E\leq 0$，所以网络的能量将会越来越小，最后趋于稳定点 $\Delta E=0$。霍普菲尔德能量函数的物理意义是：在那些渐进稳定点的吸引域内，离吸引点越远的状态，所具有的能量越大，由于能量函数的单调下降特性，保证状态的运动方向能从远离吸引点处，不断地趋于吸引点，直到达到稳定点。

几点说明：

①能量函数是反馈网络中一个很重要的概念。根据能量函数，可以很方便地判定系统的稳定性。网络的能量值与其状态存在着一定的联系，即能量的改变对应着状态的变迁，网络的稳定状态对应于能量函数的极小点。正是这种对应关系为网络进行优化计算奠定了基础。

②能量函数与李雅普诺夫函数的区别在于：李氏函数被限定在大于零的范围内，而能量函数无此要求，但要求负向有界；李氏函数要求在零点值为零，即 $v(0)=0$，而能量函数无此要求，所以，当能量函数 E 满足 $E(v)<0, E(0)=0$ 和 $\frac{\mathrm{d}E}{\mathrm{d}t}\leq 0$ 时，霍普菲尔德能

量函数就是李雅普诺夫函数了。

③霍普菲尔德选择的能量函数，它只是保证系统稳定和渐近稳定的充分条件，而不是必要条件，其能量函数也不是唯一的。为了能够使 $\frac{\mathrm{d}E}{\mathrm{d}t} \le 0$，霍普菲尔德对设计权 w_{ij} 有一个对称性的要求，即 $w_{ij} = w_{ji}$ 和 $\frac{\mathrm{d}v_i}{\mathrm{d}u_i} > 0$ 的要求，权的对称要求与实际神经网络的实验情况并不符合，实际神经细胞之间的连接没有对称性要求。不少文章阐述了即使不满足连接权矩阵对称的条件，仍然可以达到系统的稳定。

4.4.3 能量函数与优化计算

所谓优化问题，是求解满足一定约束条件下的目标函数的极小值问题。有关优化的传统算法很多，如梯度法、单纯型法等，由于在某些情况下，约束条件的过于复杂，加上变量维数较多等诸多原因，使得采用传统算法进行的优化工作耗时过多，有的甚至达不到预期的优化结果。

霍普菲尔德能量函数是一个反映了多维神经元状态的标量函数，而且可以用简单的电路形成人工神经网络，它们的互联形成了并联计算的机制。当各参数设计合理时，由电路组成的系统的状态，可以随时间的变化，最终收敛到渐近稳定点上，并在这些稳定点上使能量函数达到极小值。以此为基础，可以人为地设计出与人工神经网络相对应的电路中的参数，把优化问题中的目标函数、约束条件与霍普菲尔德能量函数联系起来。这样，当电路运行后达到的平衡点，就是能量函数的极小点，其系统状态满足了约束条件下的目标函数的极小值，以此方式，利用人工神经网络来解决优化问题。由于人工神经网络是并行计算，其计算量不随维数的增加而发生指数性质的“爆炸”，因而特别适用于解决有此问题的优化问题。

1) 能量函数设计的一般方法

设优化目标函数为 $f(u,v)$，$u \in R^r$，为人工神经网络的状态，也是目标函数中的变量。优化的约束条件为：$g(u,v) = 0$。优化问题归结为：在满足约束的条件下，使目标函数最小。由于可以设计出等价最小的能量函数 E 为：

$$E = f(u,v) + \sum |g_i(u,v)|$$

这里，$\sum |g_i(u,v)|$ 也称为惩罚函数，因为在约束条件 $g(u,v) = 0$ 不能满足时，$\sum |g_i(u,v)|$ 的值总是大于零，造成 E 不是最小。

对于目标函数 $f(u,v)$，一般总是取一个期望值与实际值之间之差的平方或绝对值的标量函数，这样能够保证 $f(u,v)$ 总是大于零。根据霍普菲尔德能量函数的要求，只有 E 在负的方向上有界，即 $|E| < E_{\max}$，同时 $\frac{\mathrm{d}E}{\mathrm{d}t} \le 0$，则系统最后总能达到 E 最小且 $\frac{\mathrm{d}E}{\mathrm{d}t} = 0$ 的点，

此点同时又是系统稳定点，即 $\frac{\mathrm{d}u_i}{\mathrm{d}t}=0$ 的点。由于求解优化问题的 E 往往是状态 u 的函数，所以，为了求解方便，常常将 $\frac{\mathrm{d}E}{\mathrm{d}t}\leq 0$ 的条件转化为对状态求导的条件如下：

$$\frac{\partial E(u_i,v_i)}{\partial u_i}=-\frac{\mathrm{d}u_i}{\mathrm{d}t} \tag{4.19}$$

这是因为：

$$\frac{\mathrm{d}u_i}{\mathrm{d}t}=-\frac{\partial E}{\partial u_i},\qquad \frac{\mathrm{d}E}{\mathrm{d}t}=\sum_i\frac{\mathrm{d}E}{\mathrm{d}u_i}\cdot\frac{\mathrm{d}u_i}{\mathrm{d}t}=-\sum_i(\frac{\mathrm{d}u_i}{\mathrm{d}t})^2\leq 0$$

可以说条件（4.19）式是 $\frac{\mathrm{d}E}{\mathrm{d}t}\leq 0$ 的另一种表达形式。同理可得另一个等价的条件为：

$$\frac{\partial E(u_i,v_i)}{\partial v_i}=-\frac{\mathrm{d}v_i}{\mathrm{d}t}$$

所以在用霍普菲尔德能量函数求解优化问题时，首先应把问题转化为目标函数和约束条件，然后构造出能量函数，并利用条件式求出能量函数中的参数，由此得到人工神经网络的连接权值。

2）具体设计步骤

① 根据要求的目标函数，写出能量函数的第一项 $f(u,v)$；

② 根据约束条件 $g(u,v)=0$，写出惩罚函数，使其在满足约束条件时为最小，作为能量函数的第二项；

③ 加上一项 $\sum_i\frac{1}{R}\int_0^{v_i}F^{-1}(\eta)\mathrm{d}\eta$，此项是人为加上的，因为在神经元状态方程中，存在一项 $-\frac{u_i}{R}$，它是在人工神经网络的电路实现中产生的。为了使设计出的优化结果能够在电路中实现而加上此项。它是一个正值函数。在运行放大增益足够大时，此项可以忽略。

④根据能量函数 E 求出状态方程，并使下式成立：

$$\frac{\partial E}{\partial u_i}=-\frac{\mathrm{d}u_i}{\mathrm{d}t}$$

⑤根据条件与参数之间的关系，求出 w_{ij} 和 b_i $(i=1,2,\ldots,r)$；

⑥求出对应的电路参数，并进行模拟电路的实现。

4.5 用 CHNN 求解 TSP 问题

4.5.1 网络设计

霍普菲尔德在 80 年代初提出的对反馈连续网路的稳定性判别的能量函数是建立在能量基础上的。同李雅普诺夫函数一样，霍普菲尔德认为，在系统的运动过程中，其内部储存的能量随着时间的增加而逐渐减少，当运动到平衡状态时系统的能量耗尽或变得最小，那么系统自然在此平衡状态处于渐近稳定。利用能量函数的概念，可以设计连续反馈网络进行优化计算。加上连续型反馈网络与电子线路之间存在着直接对应关系，这样就可以方便地用大规模电子线路来实现网络的并行自循环的运行。通过这种并行运算，有可能解决优化计算中的“指数爆炸”问题。最典型的应用例子为 TSP 问题,即旅行商问题：一个商人要在 n 个城市中不重复地各走一遍的最短路径的求解。

为了用 CHNN 来求解 TSP 问题，必须将原问题转变成数学表达式。下面是常用的一种表示法。以城市数 N = 5，代号分别取 C1、C2、C3、C4 和 C5 为例，求解出 TSP 的解可能形式比如为：C1-C4-C2-C5-C3。采用矩阵表达，可用行表示城市，列表示位置，其解可表示为如表 4.2 所示。

表 4.2 TSP 解的一种矩阵表示形式

城市 \ 位置	1	2	3	4	5
C1	1	0	0	0	0
C2	0	0	1	0	0
C3	0	0	0	0	1
C4	0	1	0	0	0
C5	0	0	0	1	0

由表 4.2 可得，TSP 问题的可行解必须满足以下三个条件：

①每行有且只有一个 1；

②每列有且只有一个 1；

③1 的总数必须等于城市数（N)。

满足以上三个条件的解为有效解，路程最短的有效解为问题的最优解。

采用 CHNN 解 TSP 问题的一般解表达式的形式可表示如下。对于 n 个城市，需要 n^2 个神经元，神经元的输入记为：$U=[u_1,u_2,\ldots,u_{n^2}]^{\mathrm{T}}$，偏差记为：$I=[i_1,i_2,\ldots,i_{n^2}]^{\mathrm{T}}$，输出记为：$V=[v_1,v_2,\ldots,v_{n^2}]^{\mathrm{T}}$。神经元输出 V 的意义如表 4.3 所示。

硬件实现的神经元由模拟运算放大器和电容、电阻构成，运算放大器的输入输出特性用激活函数 $g_i=\frac{1}{2}(1+\tanh(\lambda u_i))$ 表示，λ 称为增益参数。各神经元之间的联接用联接矩阵 T 表示。此外，各神经元的时间常数记为 $\tau_i=R_iC_i$，可简化为统一的常数 τ。描述 CHNN 的动态方程如下：

$$\dot{U}=T\cdot V-U/\tau+I \tag{4.20}$$

$$V=[1+\tanh(\lambda\cdot U)]/2 \tag{4.21}$$

表 4.3 n 个神经元网络解出 TSP 的表现形式

城市＼排序	1	2	…	n
1	v_1	v_2	…	v_n
2	v_{n+1}	v_{n+2}	…	v_{n+3}
⋮	⋮	⋮	⋮	⋮
n	$v_{n(n-1)+1}$	$v_{n(n-1)+2}$	…	v_{n^2}

本节的主要目的是采用软件来实现 CHNN，并用其解决 TSP 问题。

为了用数字计算机仿真硬件网络的动态运行过程，必须对 U 进行积分。因此，将（4.20）式改写如下：

$$\Delta U=(T\cdot V-U/\tau+I)\cdot\Delta t \tag{4.22}$$

Δt 必须足够小，这样才能用下式计算下一时刻的输入

$$U(t+\Delta t)=U(t)+\Delta U \tag{4.23}$$

反复用（4.21）式、（4.22）式和（4.23）式进行迭代，网络最终能到达一个稳定的状态，该状态就对应着问题的一个解。而解的有效性和最优性是由所谓的能量函数来保证的。

$$E=-\frac{1}{2}\sum_{i}\sum_{\substack{j\\ j\neq i}}T_{ij}v_iv_j-\sum_{i}I_iv_i+\frac{1}{\lambda}\sum_{i}\frac{1}{R_i}\int_0^{v_i}g_i^{-1}(v)\mathrm{d}v \tag{4.24}$$

其中，$g_i=\frac{1}{2}(1+\tanh(\lambda u_i))$。

上式中，第三项是由于放大器造成的，当 λ 很大时，它的作用可以忽略，我们只要根据前两项来构造能量函数即可。针对用表 4.3 中提出的表示形式，可构造出能量函数为：

$$\begin{aligned}E=&\frac{A}{2}\sum_{x=1}^{n}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}v_{xi}v_{xj}+\frac{B}{2}\sum_{i=1}^{n}\sum_{x=1}^{n}\sum_{\substack{y=1\\ y\neq x}}^{n}v_{xi}v_{yi}+\frac{C}{2}\left(\sum_{x=1}^{n}\sum_{i=1}^{n}v_{xi}-n\right)^2\\&+\frac{D}{2}\sum_{x=1}^{n}\sum_{\substack{y=1\\ y\neq x}}^{n}\sum_{i=1}^{n}d_{xy}v_{xi}\left(v_{y,i+1}+v_{y,i-1}\right)\end{aligned} \tag{4.25}$$

其中，A,B,C,D 都是正数，其意义如下：A 称为行抑制系数，它决定行抑制的程度；B 称为列抑制系数，它决定列抑制的程度；C 称为全局抑制系数，它决定全局抑制的程度；D 称为距离抑制系数，它决定距离抑制的程度；$(d_{xy})_{n\times n}$ 是 n 个城市间的距离矩阵。

当网络运行到一个稳定状态时，网络的输入和输出应当保持不变，即

$$\Delta U = U(t+\Delta t) - U(t) = 0$$

这样，终态的判断条件可以选作：

$$\max_{1\le j\le n^2}\{\Delta u_j\}\le \varepsilon$$

ε 是一个任意小的正数。

根据（4.24）式、（4.25）式可以推导解出最优解的网络权值的联接矩阵为：

$$T_{n(x-1)+i,n(y-1)+j} = -A\delta_{xy}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{xy}) - C - Dd_{xy}(\delta_{j,i+1}+\delta_{j,i-1}) \qquad (4.26)$$

其中，x, y 表示城市；i, j 表示路径中的位置；T 是一个 n^2 维的对称方阵；

$$\delta_{ij} = \begin{cases} 1 \ , & i = j \\ 0 \ , & i \ne j \end{cases}$$

正是因为在软件设计中需要考虑这些特殊问题，所以在下面几节内容中将分别对有关问题进行讨论。

4.5.2 对终态时系统的输出向量 V 的解释

所谓对终态时的输出向量 V 进行解释是指：将表 4.3 的表示形式转换为对应的路径。路径用一个行向量 $R=[r_1, r_2, \ldots, r_n]$ 表示，如对 4 座城市的情况，一种可能的路径为：R=[2 1 4 3]，它表示实际的路线为：$2\to1\to4\to3\to2$（箭头的方向也可逆转）。

由于 $0\leqslant v_i\geqslant1$，因此采用以下的方法来对终态时的输出向量进行解释：找出表 4.3 中每行中的最大值以及其所在的列数 j，该列数 j 表示城市 x 将出现在路径的第 j 个位置上，令 $r_j = x$，从而得到路径向量。查找的顺序是每行内从左到右，对城市采用从上到下。

由此产生一个问题：这样能保证每一行的最大值都会出现在不同的列上吗？答案是否定的。一个很自然的想法是：再对列作相同的操作。然而，如果对行和列的操作不相符，就必须采取一种策略来进行取舍，这种取舍必须兼顾别的行和列，这样一来，仅仅在对结果的解释上就花费了相当长的时间，从而增加了整个算法的时间复杂度，因而这种策略是不可取的。事实上，通过对能量函数的分析可以看出，非有效解的存在是必然的。根据（4.25）式，前三项保证终态时 V 的每行、每列有且只有一个 1，并且 1 的总数等于 n。第四项的意义在于：在前三项的基础上，距离最短的路径对应着能量函数的最低点。但是由于 v_{xi} 是连续的，加上非线性项 $\frac{1}{\lambda}\sum_i \frac{1}{R_i}\int_0^{v_i} g_i^{-1}(v)\mathrm{d}v$ 的影响，使得第四项会产生一些副作用：前三项的略微增大可能导致第四项的显著减小，从而使得整个能量函数减小。也就是说，在能量空间中，能量函数不仅在有效解处存在极小值，而且在某些非有效解处也存在极小值。例如，终态时输出向量由表 4.4 所示（n = 4，暂时取 $A=B=C=D=2$）。按照取每行最大值的方法，对此结果的解释是：[1 2 4 0]，这是一个非有效解。与此邻近的一个有效解是[1 2 4 3]，其对应的终态输出向量见表 4.5。

设由表 4.4 按（4.25）式算得 E_1，按表 4.5 算得 E_2，则：

$$\Delta E = E_1 - E_2 = 1.2 - 0.8d_{13} + d_{23} - 0.7d_{34}$$

若取 $d_{13}=d_{34}=1$，$d_{23}=0.2$，则 $\Delta E = -0.1$，可见该非有效解在能量函数空间中对应的值比该有效解的对应值要小，非有效解对应的状态可以是稳定的，对应于一个局部极小值，从而可能出现在终态。

表 4.4 网络终态时输出向量情形之一

城市＼位置	1	2	3	4
1	0.8	0	0	0
2	0	1	0	0
3	0	0	0.5	0.5
4	0	0	0.7	0

表 4.5 网络终态时输出向量情形之二

城市＼位置	1	2	3	4
1	0.8	0	0	0
2	0	1	0	0
3	0	0	0	1
4	0	0	0.7	0

显然，A, B, C, D, d_{xy} 会影响非有效解的出现。距离矩阵 $(d_{xy})_{n\times n}$ 由具体问题决定，它对非有效解的影响可以通过调节 A，B，C，D 的取值来补偿。实验表明，一组合适的 A，B，C，D 参数能大大减小非有效解出现的几率。此外，由对称性可知，A，B 通常相等，这样待调参数就减少为 3 个。针对非有效解的出现，我们对终态输出的解释采用了以下的处理方法，如果终态出现了非有效解，则让系统恢复到初始状态重新运行。实际上参数的选取对获得问题的满意解十分重要，而这些参数有时因不同问题而异，尤其是对城市数 n 很敏感，这在后面将进行讨论。

4.5.3 用 CHNN 算法实现 TSP 的问题探讨

1) 软件算法内存占用量

对于 n 个城市的情况，联接矩阵 T 的维数是 $n^2\times n^2$，神经网络的输入、输出以及偏差都是 $n^2\times 1$ 的向量。可见，这种算法的内存占用量很大。同时，由于程序采用大量的矩阵运算，因此，内存占用量的增加必将导致时间用量的增加。实验显示，对于 $n\geq 15$ 的情况，求出一个解的时间相当长。

生物神经网络能在很短的时间内解决一些计算机要花费很长时间才能解决的问题，其中一个最重要的原因是并行计算的机制，另一个重要的原因是用模拟式计算而不是数字式计算。模拟式计算在精确性上不如数字式计算，但是在诸如 TSP 这类优化问题上，精确的计算似乎并不重要，它反而会增大求解的时间；模拟式计算具有的模糊性能达到同时考虑多种可能决策的效果，因此在解决诸如 TSP 这类优化问题上有很高的效率。此外，神经元之间大量的相互连接也为它的强大的计算能力提供了帮助。大量的相互连接使得原本独立的计算单元（神经元）之间建立了广泛的联系，使得原本互相独立的计算过程相互影响，共同求得问题的解。霍普菲尔德提出 CHNN 的目的就是模仿生物神经网络，设计出一种大规模相互连接的、具有并行计算能力的、由模拟元件组成的人工神经网络。这种神经网络对于 TSP 这类优化问题具有很强的求解能力，而且这种网络求解 TSP 问题所需的时间是其元件时间常数的几倍。

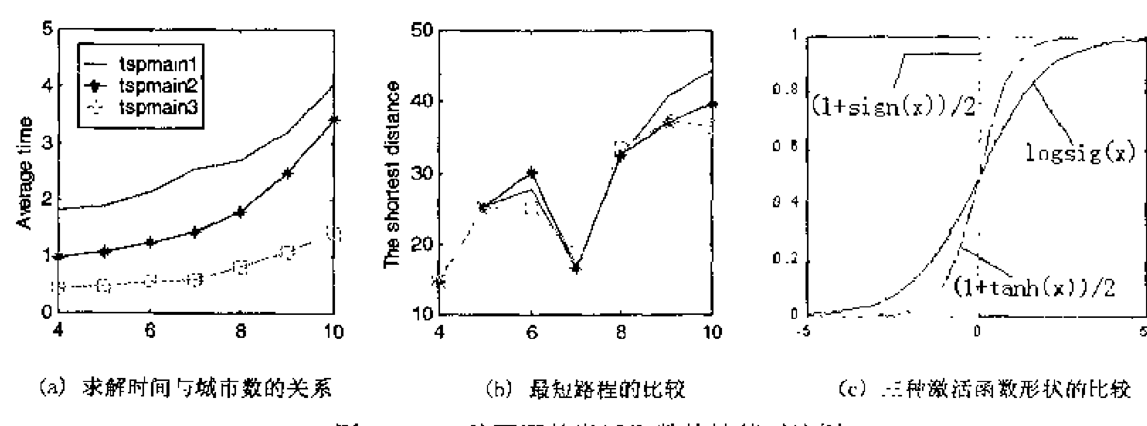
事实上用软件仿真的目的，我们认为主要有以下几点：一是验证理论的可行性，即对（4.20）式、（4.21）式所表示的人工神经网络能否求解 TSP 这类问题的检验；二是观察一些参数对求解过程的影响，从而为设计硬件电路提供一定的指导；三是利用计算机对 CHNN 作一些理论上的分析。至于软件模拟的时间和内存占用量并不具有重要的意义。原因很简单，单机上的软件模拟不具有并行计算的特点，仅具有半模拟计算的特点，因为计算机在计算的过程中采用的是数字的逻辑的方式，特别是对于积分等运算；但是，相对于别的基于搜索的优化算法而言，这种模拟的算法具有一定的“模拟性”，因为它是模拟一个动态过程从不稳定“运行”到稳定后获得解，它不像基于搜索的算法那样需要精确地计算出中间状态，从这个意义上说它具有“半模拟性”，并且从其内部的计算过程来说，它也不具有生物神经网络所具有的计算单元之间的相互作用。因此，软件仿真在时间和内存占用量上是不可少的。也许用并行计算的方法编程会有所改观。

2）在连续的状态空间内求解离散状态问题

在能量函数（4.25）中，如果 $v_{xi}=0或1$，那么在这个离散的状态空间上能量函数 E 的局部极小值对应每一个有效解，并且在最优路径上取得全局极小值。如果 $0\le v_{xi}\le 1$，那么在这个连续的状态空间上能量函数 E 的局部极小值不仅仅对应着有效解，还可能对应着一些非有效解。显然，能量函数 E 的局部极小值与有效解之间的非一一对应关系导致了问题求解过程中非有效解的出现，并且也会使求解的过程花费更多的时间。于是，一个很自然的想法是：如果将激活函数选为硬函数，那么 v_{xi} 就被限制在 0 或 1 上，这样能否减少非有效解的出现并且使求解时间缩短？答案是肯定的。下面是这方面的实验。

为了便于研究参数对求解过程的影响以及 CHNN 算法的优劣。我们所实现的主程序有 3 个版本，分别取名为：tspmain1、tspmain2、tspmain3，主要区别在于激活函数 f_i 不同，依次分别为：$f_i=\frac{1}{2}(1+\tanh(\lambda u_i))$、$f_i=\log\mathrm{sign}(\lambda u_i)$ 和 $f_i=\frac{1}{2}(1+\mathrm{sign}(u_i))$。

取城市数 n 从 3 到 10，分别运行 tspmain1、tspmain2、tspmain3，记录下它们求得一个有效解所用的时间，作出时间随 n 的变化曲线（见图 4.12）。图 4.12(a) 是三种不同的激活函数分别对应的求解时间与城市数目的关系曲线。图 4.12(b) 是三种激活函数所求得的最短路程的比较。应当注意的是，该图中只有横坐标相同的点才有可比性（单位为城市数 n），因为城市数目不同时距离矩阵也不同。图 4.12(c) 是三种激活函数形状的比较。



(a) 求解时间与城市数的关系　(b) 最短路程的比较　(c) 三种激活函数形状的比较

图 4.12 三种不同的激活函数的性能对比图

从图 4.12(a)和图 4.12(b)两幅图来看，似乎硬函数的求解效率最高，因为它所需的时间最短，且所求出解的质量也不比其他两种激活函数时的情况差。但在做这个实验时，发现当激活函数用硬函数时，程序有时会陷入死循环，而其他的激活函数则不会出现。图中还可以看出，采用硬函数后求解时间变短。当函数 $g_i=\frac{1}{2}(1+\tanh(\lambda u_i))$ 中增益参数 $\lambda\to\infty$ 后，它就等价于硬函数，这样似乎增益参数越大，求解过程越快，可见找到一个满意的增益参数对于提高求解效率很重要。但是在使用 tspmain3 的过程中，我们发现求得的解可能是任意的，从统计角度而言，它得到的解比其他两种方法得到的解要差，即距离大。此时对于 TSP 这种优化问题，求"最优解"的目标已被降低为求"次优解"，这是符合实际需要的。不过在这种前提下，所获得的平均距离的大小应当成为衡量算法好坏的一个重要标准。因此，求解效率是一个综合指标，它包括求解时间、所获得的平均距离的大小和非有效解出现几率三方面因素。在离散状态空间中求解 TSP 这类问题所需的时间较短，按霍普菲尔德的说法，在离散状态空间中求解得到的结果比较接近于随机生成路径的结果。在状态空间中求解 TSP 这类问题所需的时间较多，同时次优解出现的机会较多，状态的连续性类似于模糊理论中的隶属度，它使同时考虑各种情况成为可能。增益参数 λ 能控制激活函数非线性部分的斜率，因此很重要。目前我们觉得取 1～5 较好。λ=0.5，0.6，0.7，0.8，0.9，1，5，10，20，30，40，50，60，70，80，90 时对求解时间的影响见图 4.13。

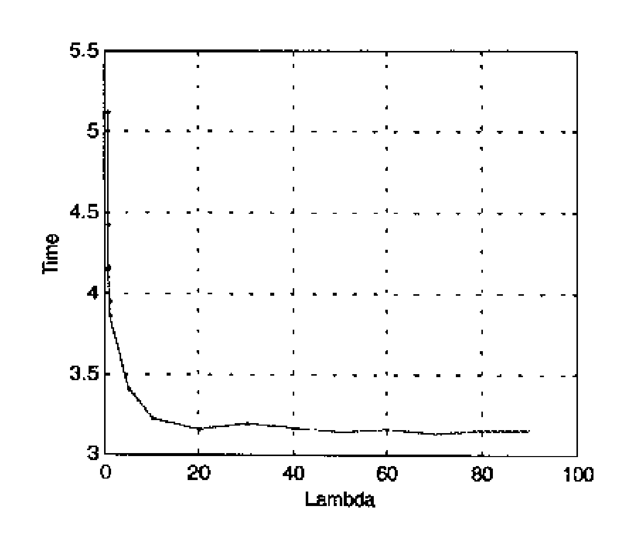


图 4.13 λ 对求解所需时间的影响

4.5.4 各参数的影响

1) *初始值的选取*

求解网络动态方程，必须有初始值 U_0。初始值的选取有以下两种方法：一是随机选取；另一种是取一常数向量 U_0，使得 $\sum v_{xi}=\sum\frac{1}{2}(1+\tanh(\lambda\cdot U_0))=n$，然后给 U_0 一个随机扰动 δu，以 $U_0+\delta u$ 作为初始值。经我们试验，这种方法比第一种方法好。通过实验，

我们将初始值定为一个常数，它具有如表 4.6(a)所示的形式，使得开始计算时网络输出 V 具有表 4.6(b)的形式。这样保证了网络开始时一定处于不稳定状态。然后再给初始值一个小扰动 $-0.1\cdot\max(U_0)\le\max(\delta U)\le 0.1\cdot\max(U_0)$，使网络运动起来。

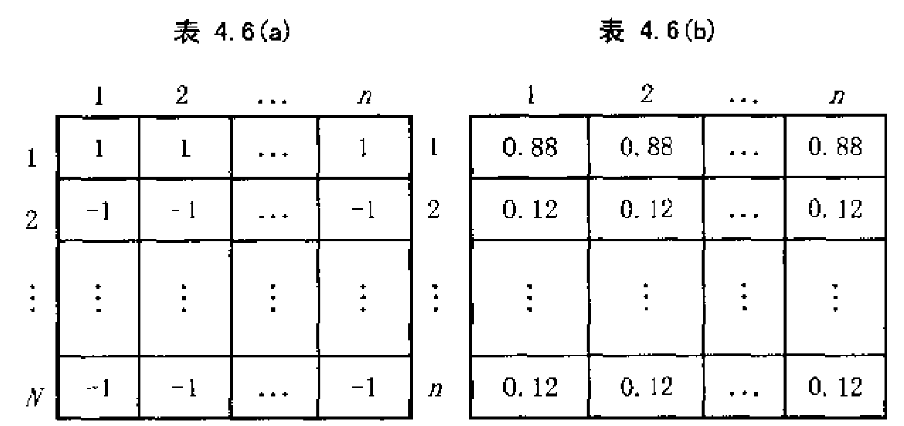


表 4.6(a)

	1	2	…	n
1	1	1	…	1
2	-1	-1	…	-1
⋮	⋮	⋮	⋮	⋮
N	-1	-1	…	-1

表 4.6(b)

	1	2	…	n
1	0.88	0.88	…	0.88
2	0.12	0.12	…	0.12
⋮	⋮	⋮	⋮	⋮
n	0.12	0.12	…	0.12

2) 偏差 I 对求解过程的影响

I 的影响主要是使能量函数空间中局部极小值处的变化率增大，也就是锐化能量函数空间。按 I 的取值计算 10 次，记下结果中有效解平均所需时间(见图 4.14(a)，纵坐标单位为秒）以及 10 次结果中非有效解个数(见图 4.14(b))，图中横坐标为 I。

由图 4.14 可见，I 取太小或太大都不好，不仅时间花费大，而且错误率高。从实验中得知，I 的可取范围为 30～50。

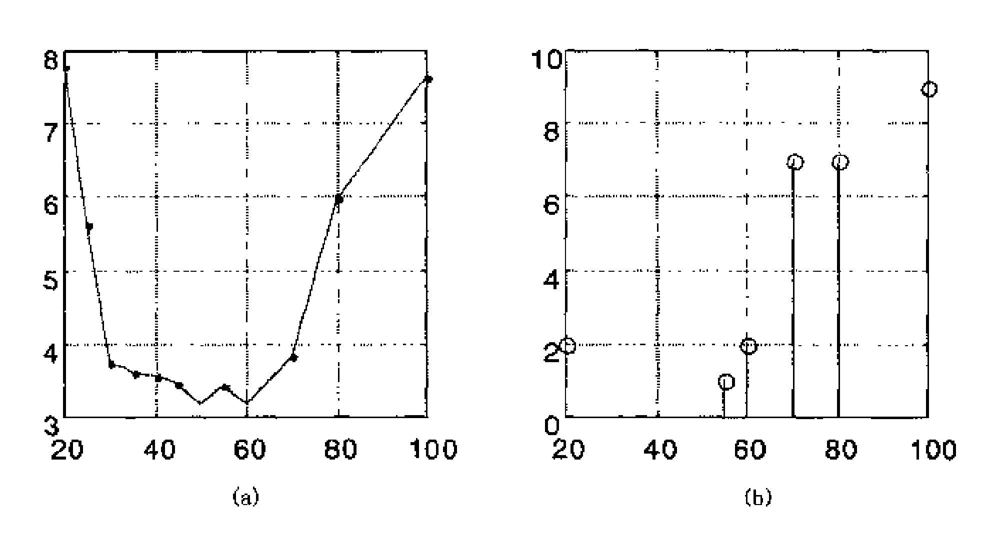


图 4.14 有效解平均所需时间次结果中非有效解个数

以上实验统一使用表 4.7 中括号外所列的参数值，是通过实验和分析获取的。

表 4.7 实验中所用各参数取值表

A, B	C	D	I	λ	τ	dT
30(30)	2(2)	6(20)	30(50)	1(1)	1(1)	0.01(0.1)

3) 各参数的选取

关于 A,B,C,D 的取值，主要是根据它们的意义来调节；此外，$A=B$，$C\le D\le A,B$

可以作为参考条件。在实验中通过实际参数调节和对比分析，我们做了对于 10 座城市进行 TSP 软件优化求解的工作，采用表 4.7 括号中所列的参数值。优化求解所得到的较好以及最好的 3 个结果如图 4.15 所示，实验数据记录在表 4.8 中。实验是在下述条件进行的，编程环境：MATLAB 5.21；操作系统：Windows 98；硬件系统：Pentium II 233 MMX，内存 128M。城市间距离矩阵和城市坐标见表 4.9 和表 4.10。

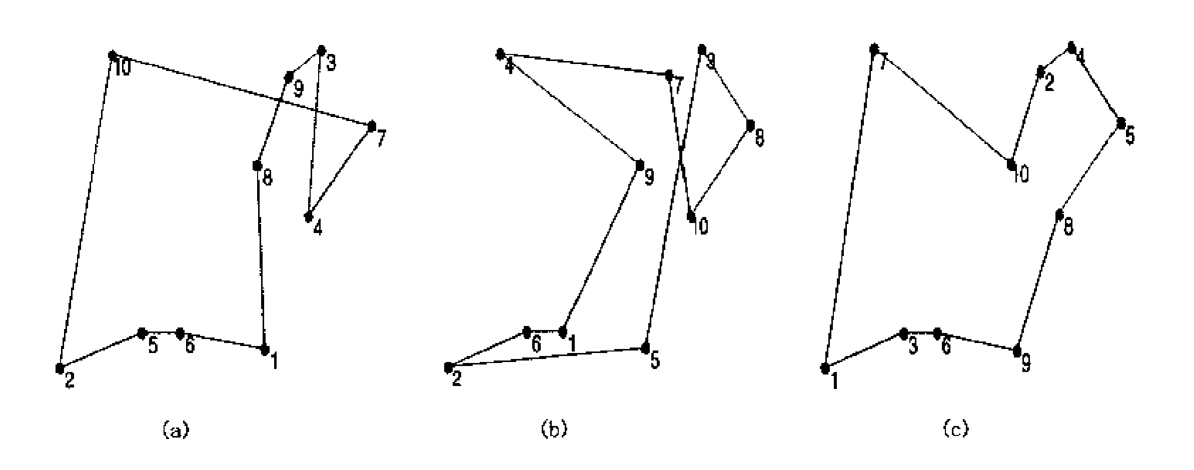


图 4.15 采用霍普菲尔德连续网络优化求解 10 座城市的 TSP 问题的结果图

表 4.8 实验数据记录

图 4.19	距离	时间（秒）	循环次数
(a)	36.3638	0.44	183
(b)	40.5606	0.43	180
(c)	30.9747	0.5	183

表 4.9 10 座城市间距离矩阵

0	8.2848	3.4172	3.66	6.1119	4.3322	7.2519	6.1982	6.4943	2.56
8.2848	0	7.0037	6.6218	7.5636	5.0608	6.2408	7.9496	5.2629	6.8594
3.4172	7.0037	0	5.7078	8.273	1.9521	3.926	2.8584	3.3794	4.7684
3.66	6.6218	5.7078	0	2.5928	5.2451	8.5529	8.4549	7.4772	1.1083
6.1119	7.5636	8.273	2.5928	0	7.6012	10.831	10.965	9.7046	3.6591
4.3322	5.0608	1.9521	5.2451	7.6012	0	3.3193	3.5682	2.3116	4.5935
7.2519	6.2408	3.926	8.5529	10.831	3.3193	0	2.3407	1.1612	7.9072
6.1982	7.9496	2.8584	8.4549	10.965	3.5682	2.3407	0	2.7933	7.5797
6.4943	5.2629	3.3794	7.4772	9.7046	2.3116	1.1612	2.7933	0	6.8916
2.56	6.8594	4.7684	1.1083	3.6591	4.5935	7.9072	7.5797	6.8916	0

表 4.10 10 座城市坐标 xy^{T}

6.5111	2.0154	7.8804	2.8723	.41456	6.3391	8.2557	9.7824	7.2538	3.9805
2.3033	9.2622	5.4342	2.696	1.8699	6.6321	9.3422	7.5679	8.755	2.6903

4.5.5 小结

总的来说，采用 CHNN 软件实现 TSP 是可行的，但存在以下一些不足：

①空间占用量大，花费时间比较多(主要是因为矩阵维数太大，计算耗时多)，因此，不能解决城市数目多的问题，对于 10 个以内的问题求解还比较理想；

②可调参数多，难以进行优化；

③可能得到非有效解。

可能的改进方法有：

①改进能量函数的形式，以避免这里遇到的一些问题，如联接矩阵维数大、存在非有效解等。已有学者提出了 TSP 问题的不同表示形式，并构造出了新的能量函数；

②采用并行的程序设计方法，实现并行计算；

③对于城市数目多的问题，可以分级计算，即先将 n 个城市的问题化为 m（$m < n$）个城市群来计算出宏观上的最优路径，再对每一个城市群进行计算，求出微观上的最优路径。应注意的是宏观和微观的算法略有不同，因为只有在最顶层才是求最短的闭合回路，其他层次上求的是两点之间的最短路径。

第 5 章 自组织竞争人工神经网络

在实际的神经网络中，比如人的视网膜中，存在着一种“侧抑制”现象，即一个神经细胞兴奋后，通过它的分支会对周围其他神经细胞产生抑制。这种侧抑制使神经细胞之间出现竞争，虽然开始阶段各个神经细胞都处于程度不同的兴奋状态，由于侧抑制的作用，各细胞之间相互竞争的最终结果是：兴奋作用最强的神经细胞所产生的抑制作用战胜了它周围所有其他细胞的抑制作用而“赢”了，其周围的其他神经细胞则全“输”了。

自组织竞争人工神经网络正是基于上述生物结构和现象形成的。它能够对输入模式进行自组织训练和判断，并将其最终分为不同的类型。与 BP 网络相比，这种自组织自适应的学习能力进一步拓宽了人工神经网络在模式识别、分类方面的应用，此外，竞争学习网络的核心——竞争层，又是许多种其他神经网络模型的重要组成部分，例如科荷伦(Kohonen)网络（又称特性图）、反传网络以及自适应共振理论网络等中均包含竞争层。

5.1 几种联想学习规则

格劳斯贝格（S. Grossberg）提出了两种类型的神经元模型：内星与外星，用以解释人类及动物的学习现象。一个内星可以被训练来识别一个矢量；而外星可以被训练来产生矢量。

由 r 个输入构成的格劳斯贝格内星模型如图 5.1 所示。

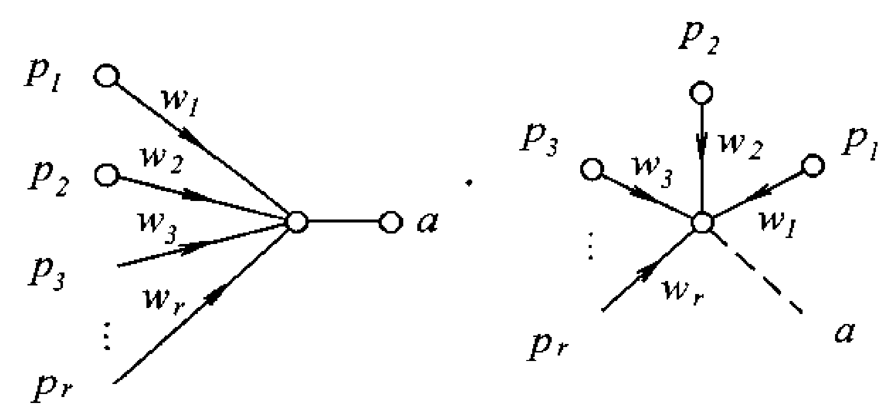


图 5.1 格劳斯贝格内星模型图

由 s 个输出节点构成的格劳斯贝格外星模型如图 5.2 所示。

从图 5.1 和图 5.2 中可以清楚地看出，内星是通过联接权矢量 W 接受一组输入信号 P；而外星则是通过联接权矢量向外输出一组信号 A。它们之所以被称为内星和外星，主要是因为其网络的结构像星形，且内星的信号流向星的内部；而外星的信号流向星的外部。下面分别详细讨论两种神经元模型的学习规则及其功效。

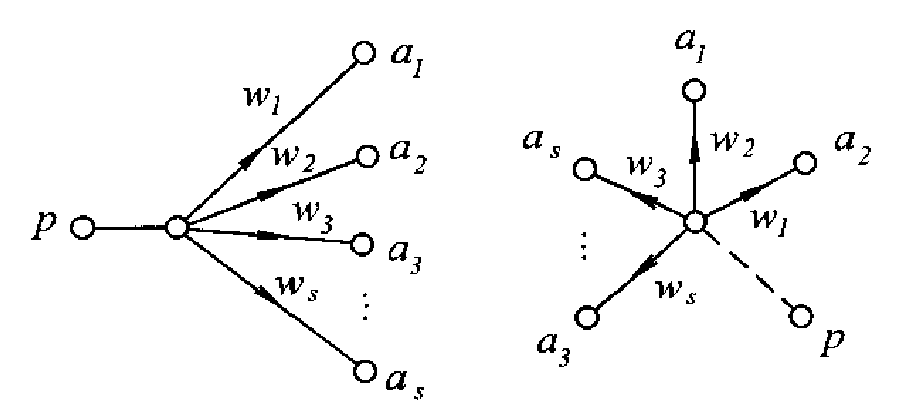


图 5.2 格劳斯贝格外星模型图

5.1.1 内星学习规则

实现内星输入/输出转换的激活函数是硬限制函数。可以通过内星及其学习规则来训练某一神经元节点只响应特定的输入矢量 P，它是借助于调节网络权矢量 W 近似于输入矢量 P 来实现的。

在图 5.1 所示的单内星中对权值修正的格劳斯贝格内星学习规则为：

$$\Delta w_{1j} = \mathrm{lr} \cdot (p_j - w_{1j}) \cdot a \text{ , } j = 1, 2, \ldots, r \tag{5.1}$$

由（5.1）式可见内星神经元联接强度的变化。Δw_{1j} 是与输出成正比的。如果内星输出 a 被某一外部方式设置而维护高值时，那么通过不断反复地学习，权值将能够逐渐趋近于输入矢量 p_j 的值，并趋使 Δw_{1j} 逐渐减少，直至最终达到 $w_{1j} = p_j$，从而使内星权矢量学习了输入矢量 P，达到了用内星来识别一个矢量的目的。另一方面，如果内星输出保持为低值时，网络权矢量被学习的可能性较小，甚至不能被学习。

现在来考虑当不同的输入矢量 P^1 和 P^2 分别出现在同一内星时的情况。首先，为了训练的需要，必须将每一输入矢量都进行单位归一化处理，即对每一个输入矢量 P^q（$q = 1, 2$），用 $1\Big/\sqrt{\sum_{j=1}^{r}(p_j^q)^2}$ 去乘以每一个输入元素，因此所得的用来进行网络训练的新输入矢量具有单位 1 的模值。

当第一个矢量 P^1 输入给内星后，网络经过训练，最终达到 $W = (P^1)^{\mathrm{T}}$。此后，给内星输入另一个输入矢量 P^2，此时内星的加权输入和为新矢量 P^2 与已学习过矢量 P^1 的点积，即

$$\begin{aligned} N &= W \cdot P^2 \\ &= (P^1)^T \cdot P^2 = \left\|P^1\right\|\left\|P^2\right\|\cos\theta_{12} = \cos\theta_{12} \end{aligned} \tag{5.2}$$

因为输入矢量的模已被单位化为 1，所以内星的加权输入和等于输入矢量 P^1 和 P^2 之间夹角的余弦。

根据不同的情况，内星的加权输入和可分为如下几种情况：

① P^2 等于 P^1，即有 $\theta_{12}=0$，此时，内星加权输入和为 1；

② P^2 不等于 P^1，随着 P^2 向着 P^1 离开方向的移动，内星加权输入和将逐渐减少，直到 P^2 与 P^1 垂直，即 $\theta_{12}=90^{\circ}$ 时，内星加权输入和为 0；

③当 $P^2=-P^1$，即 $\theta_{12}=180^{\circ}$ 时，内星加权输入和达到最小值-1。

由此可见，对于一个已训练过的内星网络，当输入端再次出现该学习过的输入矢量时，内星产生 1 的加权输入和；而与学习过的矢量不相同的输入出现时，所产生的加权输入和总是小于 1。如果将内星的加权输入和送入到一个具有略大于-1 偏差的二值型激活函数，对于一个已学习过或接近于已学习过的矢量输入，同样能够使内星的输出为 1，而其他情况下的输出均为 0。所以在求内星加权输入和公式中的权值 W 与输入矢量 P 的点积，反映了输入矢量与网络权矢量之间的相似度，当相似度接近 1 时，表明输入矢量 P 与权矢量相似，并通过进一步学习，能够使权矢量对其输入矢量具有更大的相似度，当多个相似输入矢量输入内星，最终的训练结果是使网络的权矢量趋向于相似输入矢量的平均值。

内星网络中的相似度是由偏差 b 来控制的，由设计者在训练前选定，典型的相似度值为 $b=-0.95$，这意味着输入矢量与权矢量之间的夹角小于 18° 48′。若选 $b=-0.9$ 时，则其夹角扩大为 25° 48′。

一层具有 s 个神经元的内星，可以用相似的方式进行训练，权值修正公式为：

$$\Delta w_{ij}=\text{lr}\cdot(p_j-w_{ij})\cdot a \tag{5.3}$$

一层 s 个内星神经元可以作为一个 r 到 1 的解码器。另外，内星通常被嵌在具有外星或其他元件构成的大规模网络中，以达到某种特殊的目的。内星网络常用在竞争网络中。在那里，其网络的期望输出是通过竞争网络的竞争而得到的。

5.1.2 外星学习规则

外星网络的激活函数是线性函数，它被用来学习回忆一个矢量，其网络输入 P 也可以是另一个神经元模型的输出。外星被训练来在一层 s 个线性神经元的输出端产生一个特别的矢量 A，所采用的方法与内星识别矢量时的方法极其相似。

对于一个外星，其学习规则为：

$$\Delta w_{i1}=\text{lr}\cdot(a_i-w_{i1})\cdot p_j \tag{5.4}$$

与内星不同，外星联接强度的变化 ΔW 是与输入矢量 P 成正比的。这意味着当输入矢量被保持高值，比如接近 1 时，每个权值 w_{i1} 将趋于输出 a_i 值，若 $p_j=1$，则外星使权值产生输出矢量。

当输入矢量 p_j 为 0 时，网络权值得不到任何学习与修正。

当有 r 个外星相并联，每个外星与 s 个线性神经元相连组成一层外星时，每当某个外星的输入节点被置为 1 时，与其相连的权值到矢量 w_{ij} 就会被训练成对应的线性神经元的输出矢量 A，其权值修正方式为：

$$\Delta W=\text{lr}\cdot(A-W)\cdot P \tag{5.5}$$

其中，$W=s\times r$ 为权值列矢量；lr 为学习速率；$A=s\times q$ 为外星输出。

内星与外星之间的对称性是非常有用的。对一组输入和目标来训练一个内星层与将其输入与目标相对换，来训练一个外星层的结果是相同的，即它们之间的权矩阵的结果是相互转置的。

5.1.3 科荷伦学习规则

科荷伦 (Kohonen) 学习规则是由内星规则发展而来的。对于其值为 0 或 1 的内星输出，当只对输出为 1 的内星权矩阵进行修正，即学习规则只应用于输出为 1 的内星上，将内星学习规则中的 a_i 取值 1，则可导出科荷伦规则为：

$$\Delta w_{ij} = \text{lr} \cdot (p_j - w_{ij}) \tag{5.6}$$

科荷伦学习规则实际上是内星学习规则的一个特例，但它比采用内星规则进行网络设计要节省更多的学习，因而常常用来替代内星学习规则。

一般情况下，科荷伦学习规则比内星学习规则能够提高训练速度 1 到 2 个数量级。

5.2 自组织竞争网络

5.2.1 网络结构

竞争网络由单层神经元网络组成，其输入节点与输出节点之间为全互联结。因为网络在学习中的竞争特性也表现在输出层上，所以在竞争网络中把输出层又称为竞争层，而与输入节点相连的权值及其输入合称为输入层。实际上，在竞争网络中，输入层和竞争层的加权输入和共用同一个激活函数，如图 5.3 所示。

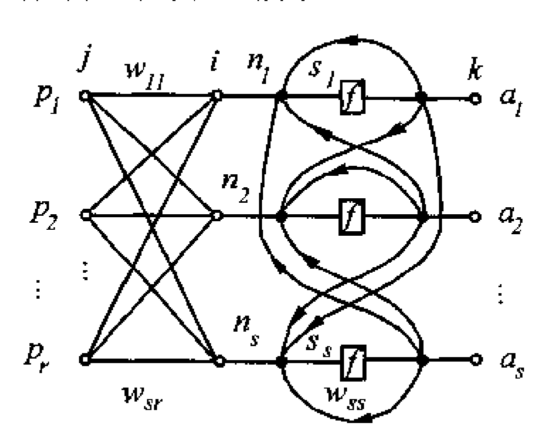


图 5.3 竞争网络结构图

竞争网络的激活函数为二值型{0,1}函数。

从网络的结构图中可以看出，自组织竞争网络的权值有两类：一类是输入节点 j 到 i 的权值 w_{ij} ($i = 1, 2, ..., s$; $j = 1, 2, ..., r$)，这些权值是通过训练可以被调整的；另一类是竞

争层中互相抑制的权值 w_{ik} ($k = 1, 2, ..., s$)。这类权值是固定不变的，且它满足一定的分布关系，如距离近的抑制强，距离远的抑制弱。另外，它们是一种对称权值，即有 $w_{ik} = w_{ki}$，同时相同神经元之间的权值起加强的作用，即满足 $w_{11}=w_{22}=...= w_{kk} >0$，而不同神经元之间的权值相互抑制，对于 $k \neq i$，有 $w_{ij}<0$。

下面来具体分析竞争网络的输出情况。

设网络的输入矢量为：$P = [\, p_1 p_2 \ldots p_r \,]^{\mathrm{T}}$ ；

对应网络的输出矢量为：$A = [\, a_1 a_2 \ldots a_s \,]^{\mathrm{T}}$。

由于竞争网络中含有两种权值，所以其激活函数的加权输入和也分为两部分：一为来自输入节点的加权输入和 N，一为来自竞争层内互相抑制的加权输入和 G。具体地说，对于第 i 个神经元，有：

①来自输入节点的加权输入和为：

$$n_i = \sum_{j=1}^{r} w_{ij} \cdot p_j \tag{5.7}$$

②来自竞争层内互相抑制的加权输入和为：

$$g_i = \sum_{k \in D} w_{ik} \cdot a_k \tag{5.8}$$

这里，D 表示竞争层中含有神经元节点的某个区域，如果 D 表示的是整个竞争层，竞争后只能有一个神经元兴奋而获胜；如果竞争层被分成若干个区域，则竞争后每个区域可产生一个获胜者。

由于 g_i 与网络的输出值 a_i 有关，而输出值又是由网络竞争后的结果所决定的，所以 g_i 的值也是由竞争结果确定的。为了方便起见，下面以 D 这整个网络输出节点的情况来分析竞争层内互相抑制的加权输入和 g_i 的可能结果。

a) 如果在竞争后，第 i 个节点“赢”了，则有：

$$a_i = 1, \quad k = i$$

而其他所有节点的输出均为零，即：

$$a_k = 0, \quad k = 1, 2, \ldots, s; \quad k \neq i$$

此时

$$g_i = \sum_{k=1}^{s} w_{ik} \cdot a_k = w_{ii} > 0 \tag{5.9}$$

b) 如果在竞争后，第 i 个节点“输”了，而“赢”的节点为 l，则有：

$$a_l = 1, \quad k = l$$

$$a_k = 0, \quad k = 1, 2, \ldots, s; \quad k \neq l$$

此时

$$g_i = \sum_{k=1}^{s} w_{ik} \cdot a_k = w_{il} < 0 \tag{5.10}$$

所以对整个网络的加权输入总和有下式成立：

$s_l = n_l + w_{ll}$，　对于“赢”的节点 l；

$s_i = n_i - |w_{il}|$，　对于所有“输”的节点 $i = 1, 2, ..., s, i \neq l$ 。

由此可以看出，经过竞争后只有获胜的那个节点的加权输入总和为最大。竞争网络的输出为：

$$a_k = \begin{cases} 1 & s_k = \max(s_i, i = 1,2,\ldots,s) \\ 0 & \text{其他} \end{cases} \tag{5.11}$$

因为在权值的修正过程中只修正输入层中的权值 w_{ij}，竞争层内的权值 w_{ik} 是固定不变的，它们对改善竞争的结果只起到了加强或削弱作用，即对获胜节点增加一个正值，使其更易获胜，而对输出的节点增加一个负值，使其更不易获胜，而对改变节点竞争结果起决定性作用的还是输入层的加权和 n_i，所以在判断竞争网络节点胜负的结果时，可直接采用 n_i，即：

$$n_{赢} = \max(\sum_{j=1}^{r} w_{ij} p_j) \tag{5.12}$$

取偏差 B 为零是判定竞争网络获胜节点时的典型情况，偶尔也采用下式进行竞争结果的判定：

$$n_{赢} = \max(\sum_{j=1}^{r} w_{ij} p_j + b), \quad -1 < b < 0 \tag{5.13}$$

典型的 b 值取-0.95。加上 b 值意味着取 $b = -|w_{ij}|$ 这一最坏的情况。

通过上面分析，可以将竞争网络的工作原理总结如下：竞争网络的激活函数使加权输入和为最大的节点赢得输出为 1，而其他神经元的输出皆为 0。

5.2.2 竞争学习规则

竞争网络在经过竞争而求得获胜节点后，则对与获胜节点相连的权值进行调整，调整权值的目的是为了使权值与其输入矢量之间的差别越来越小，从而使训练后的竞争网络的权值能够代表对应输入矢量的特征，把相似的输入矢量分成了同一类，并由输出来指示所代表的类别。

竞争网络修正权值的公式为：

$$\Delta w_{ij} = \text{lr} \cdot (p_j - w_{ij}) \tag{5.14}$$

式中，lr 为学习速率，且 $0 < \text{lr} < 1$，一般的取值范围为 0.01 ~ 0.3； p_j 为经过归一化处理后的输入。

不论采用哪种学习方法，层中每个最接近输入矢量的神经元，通过每次权值调整而使权值矢量逐渐趋于这些输入矢量。从而竞争网络通过学习识别了在网络输入端所出现的矢量，并将其分为某一类。

5.2.3 竞争网络的训练过程

弄懂网络的训练过程是为了更好地设计出网络。

因为只有与获胜节点相连的权值才能得到修正，通过其学习法则使修正后的权值更加接近其获胜输入矢量。结果是，获胜的节点对将来再次出现的相似矢量（能被偏差 b 所包容，或在偏差范围以内的)更加容易赢得该节点的胜利。而对于一个不同的矢量出现时，就更加不易取胜，但可能使其他某个节点获胜，归为另一类矢量群中。随着输入矢量的重复出现而不断地调整与胜者相连的权矢量，以使其更加接近于某一类输入矢量。最终，如果有足够的神经元节点，每一组输入矢量都能使某一节点的输出为 1 而聚为该类。通过重复训练，自组织竞争网络将所有输入矢量进行了分类。

所以竞争网络的学习和训练过程，实际上是对输入矢量的划分聚类过程，使得获胜节点与输入矢量之间的权矢量代表获胜输入矢量。

这样，当达到最大循环的值后，网络已重复多次训练了 P 中的所有矢量，训练结束后，对于用于训练的模式 P，其网络输出矢量中，其值为 1 的代表一种类型，而每类的典型模式值由该输出节点与输入节点相连的权矢量表示。

竞争网络的输入层节点 r 是由已知输入矢量决定的，但竞争层的神经元数 s 是由设计者确定的，它们代表输入矢量可能被化为的种类数，其值若被选得过少，则会出现有些输入矢量无法被分类的不良结果；但若被选得太大，竞争后可能有许多节点都被空闲，而且在网络竞争过程中还占用了大量的设计量和时间，在一定程度上造成了一定的浪费，所以一般情况下，可以根据输入矢量的维数及其估计，再适当地增加些数目来确定。

另外还要事先确定的数有：学习速率和最大循环数。竞争网络的训练是在达到最大循环次数后停止，这个数一般可取输入矢量数组的 40~50 倍，即使每组输入矢量能够在网络重复出现 40~50 次。

然后网络则可以进入竞争以及权值的调整阶段。

竞争网络比较适合用于具有大批相似数组的分类问题。下面给出例题来说明竞争网络的功效。

例 5.1　对下列模式 P 进行分类辨识。

P = [0.7071 0.6402 0.000 -0.1961 0.1961 -0.9285 -0.8762 -0.8192 ;
    [0.7071 0.7682 -1.000 -0.9806 -0.9806 0.3714 0.4819 0.5735];

解：　输入模式 P 已经为归一化处理后的数据。对于网络结构，我们取 $S = 4$；并随机取一组初始矩阵为：

```
W0 = [ -0.3347 -0.9413;
       -0.5466 -0.8374;
       -0.8690 -0.4948;
        0.4611 -0.8874];
```

另取：

```
lr = 0.05;
max_epoch = 320;
```

这里取了输入数据的20倍。最大循环数一般应根据输入数据的多少来决定。

1）训练竞争

为了能够更清楚地理解训练竞争过程，我们仍可以通过图解法来解释。和感知器的分类方式类似，二值型分类的实质是通过将输入矢量空间进行分割而达到分类目的。在此，我们用横、纵坐标分别表示 p_1 和 p_2 矢量。与感知器的不同的是，竞争网络所用的矢量是模为 1，所以矢量的变化，在坐标上形成的轨迹为以原点为中心的单位圆。网络竞争的目的，是使权值 W 经过竞争后逐渐移动到能够代表输入矢量类别的点上（也处于单位圆上）。

在训练过程中，当第一次出现输入矢量比如 P^l 时，有 $W_{lj}=p^1$, $l\in S$, $j=1,2,\ldots,r$。但当 P^2 出现时，若也具有与 W_{lj} 相似的特性。则通过对 W_{lj} 的修正，使 W_{lj} 倾向 P^2，当下一次 P^l 再次输入时，W_{lj} 又被修正得移向 P^1。这样，经过多次训练后，W_{lj} 所得的结果为几个相同类型输入模式的平均值，而当一个不同类型的模式输入，将使竞争网络中的其他节点获胜而得到一个新的权矩阵。如此重复，可以将所有的不同类型的模式都聚集到相同的权矩阵下，每个权矩阵代表一种类型，而输出矢量中的 1 的列位置则指出与此节点相连的权矢量所代表的输入矢量组的位置。图 5.4 为网络竞争前的权矢量位置图，图 5.5 给出最后的训练结果，图中权矢量位置是用“o”来表示的。

本例题经过 320 次循环后得到最后结果为：

```
W= [ -0.0460 -0.9863;
     -0.5466 -0.8374;
     -0.8748  0.4747;
      0.6684  0.7180];
```

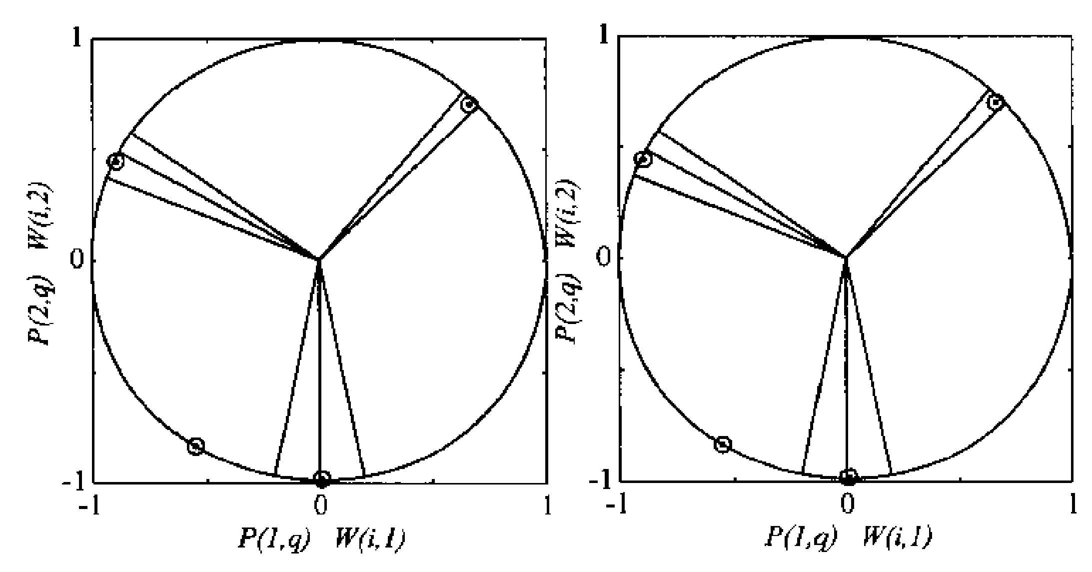


图 5.4　竞争前的权矢量位置图　　图 5.5　竞争结束后的权矢量位置图

在输入模式 P 作用下的网络输出为：

```
A = [0 0 1 1 1 0 0 0;   — 1 类 （第 3，4，5 列输入矢量组属于同一类）
     0 0 0 0 0 0 0 0;   — 空缺
     0 0 0 0 0 1 1 1;   —3 类 （第 6，7，8 列输入矢量组属于同一类）
     1 1 0 0 0 0 0 0];  —4 类 （第 1，2 列输入矢量组属于同一类）
```

将 W 值与前面 $W0$ 相比较可以看出，第二行的值与初始 $W0$ 值完全相同，即在整个竞争训练中，W_{2j} 从未获得过胜利，所以其权矩阵一次也没有得到过修正，这从 A 的输出值上看得很清楚：当 P^1、P^2 输入时，输出节点 4 获胜，即它们属于 W_{4j} 类；W_{4j} 可代表它们的值，P^3、P^4 和 P^5 属第 1 类，W_{1j} 是他们三个的代表，P^6、P^7 和 P^8 同属第 3 类，W_{3j} 反映了它们的特点。

2）竞争学习网络的局限性

竞争网络适用于具有典型聚类特性的大量数据的辨识，但当遇到大量的具有概率分布的输入矢量时，竞争网络就无能为力了，这时可以采用科荷伦网络来解决。

5.3 科荷伦自组织映射网络

神经细胞模型中还存在着一种细胞聚类的功能柱。它是由多个细胞聚合而成的，在接受外界刺激后，它们会自动形成。一个功能柱中的细胞完成同一种功能。

生物细胞中的这些现象在科荷伦网络模型中有所反映。当外界输入不同的样本到科荷伦自组织映射网络中，一开始时输入样本引起输出兴奋的位置各不相同，但通过网络自组织后会形成一些输出群，它们分别代表了输入样本的分布，反映了输入样本的图形分布特征，所以科荷伦网络常常被称为特性图。

科荷伦网络使输入样本通过竞争学习后，功能相同的输入靠得比较近，不同的分得比较开，以此将一些无规则的输入自动排开，在联结权的调整过程中，使权的分布与输入样本的概率密度分布相似。所以科荷伦网络可以作为一种样本特征检测器，在样本排序、样本分类以及样本检测方面有广泛的应用。

一般可以这样说，科荷伦网络的权矢量收敛到所代表的输入矢量的平均值，它反映了输入数据的统计特性。再扩大一点，如果说一般的竞争学习网络能够训练识别出输入矢量的点特征，那么科荷伦网络能够表现出输入矢量在线上或平面上的分布特征。

当随机样本输入到科荷伦网络时，如果样本足够多，那么在权值分布上可以近似于输入随机样本的概率密度分布，在输出神经元上也反映了这种分布，即概率大的样本集中在输出空间的某一个区域，如果输入的样本有几种分布类型，则它们各自会根据其概率分布集中到输出空间的各个不同的区域。每一个区域代表同一类的样本，这个区域可以逐步缩少，使区域的划分越来越明显。在这种情况下，不论输入样本是多少维的，都可以投影到低维的数据空间的某个区域上。这种形式也称为数据压缩。同时，如果高维空间中比较相近的样本，则在低维空间中的投影也比较相近，这样就可以从中取出样本空间中较多的信息。遗憾的是，网络在高维映射到低维时会出现畸变，且压缩比越大，畸变越大；另外，网络要求的输入节点数很大，因而科荷伦网络比其他人工神经网络（如 BP 网络）的规模要大。

5.3.1 科荷伦网络的拓扑结构

科荷伦网络结构也是两层：输入层和竞争层。与基本竞争网络不同之处是其竞争层可以由一维或二维网络矩阵方式组成，且权值修正的策略也不同。

①一维网络结构与基本竞争学习网络相同；

②二维网络结构，如图 5.6 所示，网络上层有输出节点 s 个，按二维形式排成一个结点矩阵，输入节点处于下方，有 r 个矢量，即 r 个节点，所有输入节点到所有输出节点之间都有权值连接，而且在二维平面上的输出结点相互间也可能是局部连接的。

科荷伦网络的激活函数为二值型函数。一般情况下 b 值固定，其学习方法与普通的竞争学习算法相同。在竞争层中，每个神经元都有自己的邻域，图 5.7 为一个在二维层中的主神经元。主神经元具有在其周围增加直径的邻域。一个直径为 1 的邻域包括主神经及它的直接周围神经元所组成的区域；直径为 2 的邻域包括直径 1 的神经元以及它们的邻域。图中主神经元的位置是通过从左上端第一列开始顺序从左到右、从上到下找到的。如图中的 10×10 神经元层，其主神经元位于 65。

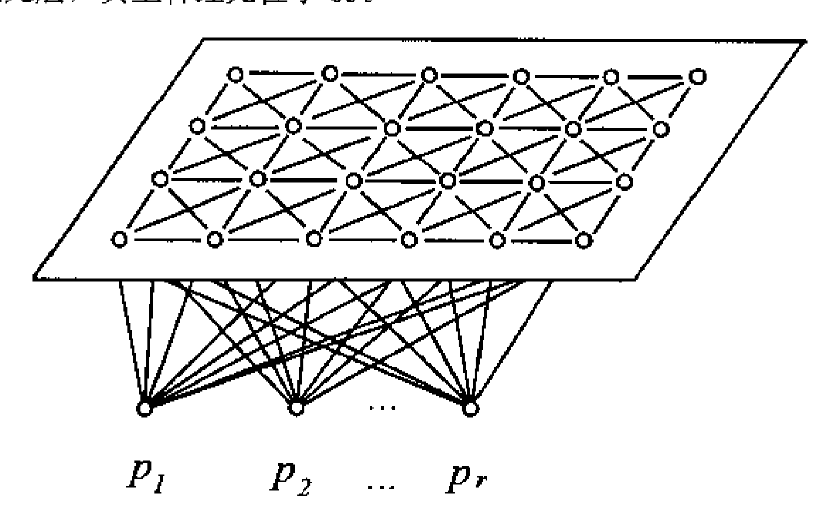
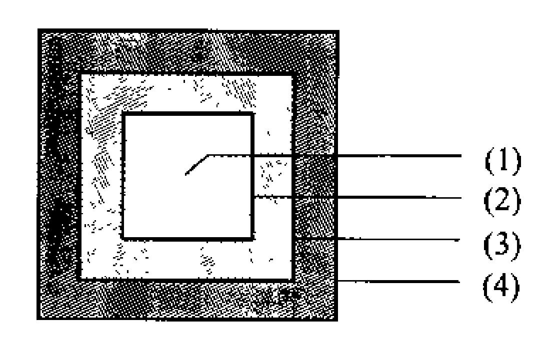


图 5.6 二维科荷伦网络结构图



(1)— 主神经元；(2)— 邻层 1；(3)— 邻层 2；(4)— 邻层 3

图 5.7 二维神经元层示意图

特性图的激活函数也是二值型函数，同竞争网络一样可以取偏置 b 为零或固定为一常数。竞争层的竞争结果，不仅使加权输入和为最大值者获胜而输出为 1，同时也使获胜节点周围的邻域也同时输出为 1。另外，在权值调整的方式上，特性图网络不仅调整与获胜节点相连的权值，而且对获胜节点邻域节点的权值也进行调整，即使其周围 D_k 的区域内神经元在不同程度上也得到兴奋，在 D_k 以外的神经元都被抑制，这个 D_k 区域可以是以获胜节点为中心的正方形，也可以为六角形。对于一维输出，D_k 则为以 k 为中心的下上邻点。

5.3.2 网络的训练过程

科荷伦网络在训练开始时和普通的竞争网络一样，其输入节点竞争的胜利者代表某类模式。然后定义获胜节点的邻域节点，即以获胜节点为中心的某一半径内的所有节点，并对与其相似的权矩阵进行调整。随着训练的继续进行，获胜节点 k 的半径将逐渐变小，直到最后只包含获胜节点 k 本身。也就是说，在训练的初始阶段，不但对获胜的节点作权值的调整，而且对其周围较大范围内的几何邻接节点也作相应的调整，而随着训练过程的进行，与获胜输出节点相连的权矩阵就越来越接近其所代表的模式类，此时，需要对获胜节点进行较细致的权矩阵调整。同时，只对其几何邻接较接近的节点进行相应的调整。这样，在训练结束后，几何上相近的输出节点所连接的权矢量既有联系(即类似性)，又相互有区别，保证了对于某一类输入模式，获胜节点能作出最大的响应，而相邻节点作出较少响应。几何上相邻的节点代表特征上相似的模式类别。

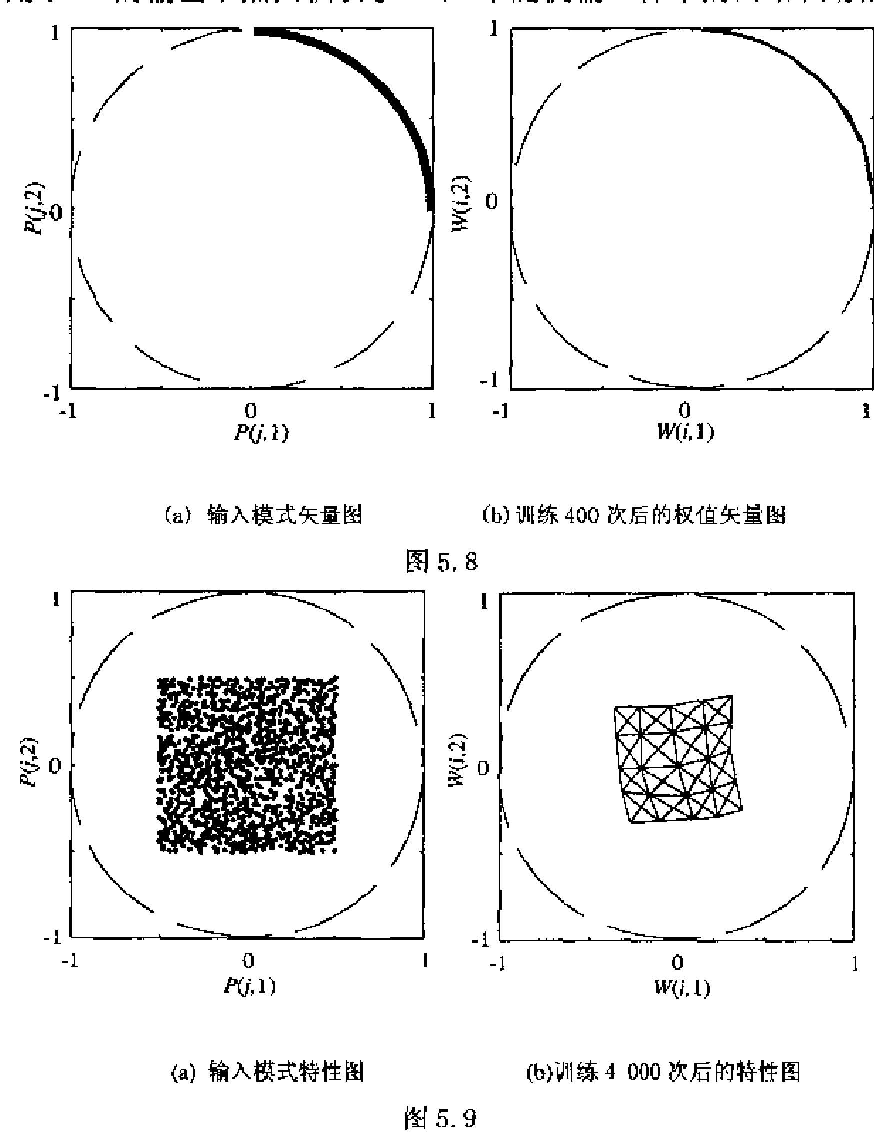
5.3.3 科荷伦网络的应用

当大量具有典型特性的模式需要进行分类时，除了可以用感知器进行分类外，还可以采用自组织竞争网络，利用竞争网络进行模式分类的另一个特点是，它不但可以将每个模式分成各自所属于的类型，而且还可以同时获得每一种类型的典型模式（一般为该类型模式的平均值）。从网络结构上讲，竞争网络是一种单层神经网络，与普通单层网络不同的是，除了输入节点与输出节点之间为全面联接外，竞争网络在输出层每个输出节点之间还具有相互的联接，并且网络在学习中的竞争特性也是表现在输出层上，所以在竞争网络中，人们把输出层又称为竞争层，而输入节点及与其相连的权值合称为输入层。实际上，在竞争层中，输入层与竞争层的加权输入和共用同一个激活函数，其激活函数的类型为阶跃型{0，1}函数。网络的每次输入只有唯一的一个输出节点在经过竞争获胜后能够输出为 1，而所有其他输出均为 0。

科荷伦网络可以作为一种样本特性检测器，在样本排序、样本分类以及样本检测方面有广泛的应用。在无监督式学习规则下，网络通过重复几十次的样本输入并不断修正网络权值，最终使网络的权值收敛到所代表的输入模式的平均值，它反映了输入数据（模式）的统计特性以及统计密度的大小。

图 5.8 为落在单位圆四分之一周期圆周上随机的 100 个点在经过 20 个神经元的科荷伦网络将其表示出的最后结果。图中各种神经元之间的距离长短表示其密度的大小，距离越短，密度越高。

图 5.9 为用 5×5 的输出节点面积表示 2000 个随机输入样本的网络训练后的特性图。



(a) 输入模式矢量图　　(b) 训练 400 次后的权值矢量图

图 5.8

(a) 输入模式特性图　　(b) 训练 4 000 次后的特性图

图 5.9

5.4 小　结

由以上两个例子可以看出：

①科荷伦自组织映射网络除可以进行分类外，还具有数据压缩功能；

②包括竞争网络在内，这些网络的工作输入模式都必须经过归一化处理，即将输入模式转化为单位矢量来处理；

③这些网络的各自特点是：普通竞争网络能够训练识别出输入模式所在的“点”特征；科荷伦网络能够表现输入模式在“线”上或“平面”上的分布特性。

第 6 章 径向基函数网络

6.1 径向基函数及其网络分析

径向基函数（Radial Basis Function，简称 RBF）网络是在借鉴生物局部调节和交叠接受区域知识的基础上提出的一种采用局部接受域来执行函数映射的人工神经网络。RBF 网络结构是由一个隐含层（径向基层）和一个线性输出层组成的前向网络，径向基层的结构图如图 6.1 所示。隐含层采用径向基函数作为网络的激活函数，径向基函数是一个高斯型函数，它是将该层权值矢量 W 与输入矢量 P 之间的矢量距离与偏差 b 相乘后作为网络激活函数的输入。

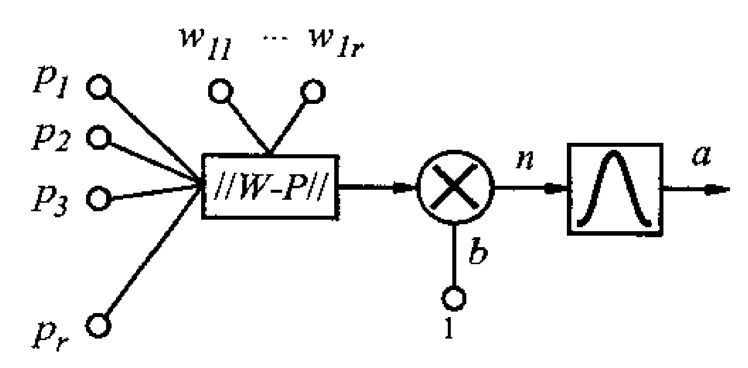


图 6.1 具有 R 个输入节点的径向基函数网络结构图

由图 6.1 可知，径向基层输入的数学表达式为 $n=\sqrt{\sum(w_i-p_i)^2}*b$，径向基层输出的数学表达式为：

$$\begin{aligned} a &= \mathrm{e}^{-n^2} = \mathrm{e}^{-(\sqrt{\sum(w_i-p_i)^2}*b)^2} \\ &= \mathrm{e}^{-(\|W-P\|\cdot b)^2} \end{aligned} \tag{6.1}$$

由（6.1）式可以看出，随着 W 和 P 之间距离的减少，径向基函数输出值增加，且在其输入为 0 时，即 W 和 P 之间的距离为 0 时，输出为最大值 1。由此，可以将一个径向基神经元作为一个当其输入矢量 P 与其权值矢量 W 相同时输出为 1 的探测器。

径向基层中的偏差 b 可以用来调节基函数的灵敏度，不过在实际应用中，更直接使用的是另一个称之为伸展常数 C 的参数。用它来确定每一个径向基层神经元对其输入矢量，也就是 P 与 W 之间距离响应的面积宽度。C 值（或 b 值）在实际应用中有多种确定方式。

径向基层的工作原理采用的是聚类功能。不失一般性，考虑 R 维空间的 q 个数据点(X^1，X^2，…，X^q)，假定数据已归一化到一个超立方体中。由于每个数据点都是聚类中心的候选者，因此，数据点 X^i 处的密度指标定义为：

$$D_i = \sum_{j=1}^{q} \exp\left(-\frac{\left\| X^i - X^j \right\|^2}{(A/2)^2} \right) \tag{6.2}$$

这里，A 是一个正数。显然，如果一个数据点具有多个临近的数据点，则该数据点具有高密度值。半径 A 定义了该点的一个邻域；半径以外的数据点对该点的密度指标贡献甚微。

在计算每个数据点的密度指标后，选择具有最高密度指标的数据点为第一个聚类中心，令 X_{c_1} 为选中的点，D_{c_1} 为其密度指标。那么每个数据点 X^i 的密度指标可用公式：

$$D_i = D_i - D_{c_1} \sum_{j=1}^{q} \exp\left(-\frac{\left\| X^i - X^j \right\|^2}{(C/2)^2} \right) \tag{6.3}$$

来修正。其中 C 是一个正数。显然，靠近第一个聚类中心 X_{c_1} 的数据点的密度指标将显著减小，这样使得这些点不太可能成为下一个聚类中心。常数 C 定义了一个密度指标显著减小的邻域。常数 C 通常大于 A，以避免出现相距很近的聚类中心。一般可选 $C = 1.5A$。此方法可称为减法聚类法。

径向基函数网络中的径向基层就是利用以上的聚类方法来作为径向基函数的中心计算函数的输出的。在人们常用的 MATLAB 神经网络工具箱中，b 和 C 之间的关系式设置为 $b = 0.8326/C$，将 b 值代入（6.1）式，有：

$$a = e^{-\left(\frac{\|W-P\| \cdot 0.8326}{C}\right)^2} = e^{-0.8326^2\left(\frac{\|W-P\|}{C}\right)^2} \tag{6.4}$$

此种参数选法使得当 $\|W-P\| = C$ 时，有 $a = e^{-0.8326^2} = 0.5$。

由此可见，当取 $b = 0.8326/C$，对任意给定的一个 C 值，可使激活层在加权输入的 $\pm C$ 处 RBF 的输出为 0.5；而通过调整 C 值，可使当 $\|W-P\| \le C$ 时，RBF 的输出大于或等于 0.5，从而直观地达到了调整 RBF 曲线宽度的目的。

C 与 $\|W-P\|$ 以及 RBF 输出之间的关系如图 6.2 所示，其中图 6.2（b）表示中心为 W，宽度为 C 的 RBF 曲线图。RBF 网络结构图如图 6.3 所示，它是由一个径向基层和一个线性输出层组成的。

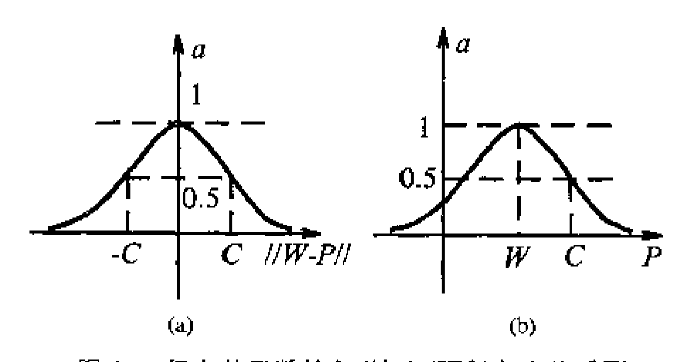


图 6.2 径向基函数输入/输出/面积宽度关系图

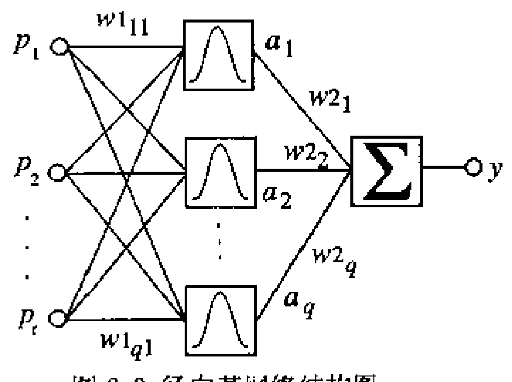


图 6.3 径向基网络结构图

6.2 网络的训练与设计

RBF 网络的训练与设计分为两步，第一步是采用非监督式的学习训练 RBF 层的权值，第二步是采用监督式学习训练线性输出层的权值，网络设计仍需要用于训练的输入矢量矩阵 P 以及目标矢量矩阵 T，另外还需要给出 RBF 层的伸展常数 C。训练的目的是为了求得两层网络的权值 $W1$ 和 $W2$ 及偏差 $b1$ 和 $b2$。

RBF 层的权值训练是通过不断地使 $w1_{ij} \to p_j^q$ 的训练方式，使该层在每个 $w1_{ij} \to p_j^q$ 处使 RBF 的输出为 1，从而当网络工作时，将任一输入送到这样一个网络时, RBF 层中的每个神经元都将按照输入矢量接近每个神经元的权值矢量的程度来输出其值。结果是，与权值相离很远的输入矢量，使 RBF 层的输出接近 0，这些很小的输出对后面的线性层的影响可以忽略。另外，任意非常接近输入矢量的权值，RBF 层将输出接近 1 的值。此值将与第二层的权值加权求和后作为网络的输出，而整个输出层是 RBF 层输出的加权求和。理论证明，只要 RBF 层有足够的神经元，一个 RBF 网络可以任意期望的精度逼近任何函数。

通常 RBF 网络的隐含层的节点数的确定是设定为与输入 P 中的样本组数 q 相同的数目，且每个 RBF 层中的权值 $W1$ 被赋予一个不同输入矢量的转置，以使得每个 RBF 神经元都作为不同 P_j^q 的探测器。对于有 q 组输入矢量，则 RBF 层中的神经元数为 q。b_1 中的每个偏差都被置为 $0.8326/C$，由此来确定输入空间中每个 RBF 响应的面积宽度。例如，C 取 4，那么每个 RBF 神经元对任何输入矢量与其对应的权值矢量之间的距离小于 4 的响应为 0.5 以上。一般而言，在 RBF 网络设计中，随着 C 取值的增大，RBF 的响应范围可以扩大，且各神经元函数之间的平滑度也较好；C 取值的减小，则使得函数形状较窄，使得与权值矢量距离较近的输入才有可能接近 1 的输出，而对其他输入的响应不敏感。所以，当采用与输入数组相同数目的 RBF 层神经元时，C 值可以取的较小（比如 $C<1$）。但当希望用较少的神经元数去逼近较多输入数组（即较大输入范围）时，应当取较大的 C 值（比如 $C=1\sim4$），以保证能使每个神经元可同时对几个输入组都有较好的响应。

在确定了$W1$和$b1$后，RBF 层的输出a_i则可求出。此时，可以根据第二层的输入a_i以及网络输出的目标T，通过使网络输出y与目标输出T的误差平方和最小来求线性输出层的权值$W2$及其偏差$b2$，不过$b2 \equiv 0$。这是因为，我们所求解的是一个具有Q个限制（输入/输出目标对）的解，而每个神经元有$Q+1$个变量（来自Q个神经元的权值以及一个偏差$b2$），一个具有Q个限制和多于Q个变量的线性方程组将有无穷多个非零解。

上述设计过程的缺点是：所设计出的 RBF 网络产生的隐含层网络所具有的神经元数目与输入矢量组数相同。当需要许多组矢量来定义一个网络时，用此方法设计的网络可能是不可接受的。对此采用的改进思想则是：在满足目标误差的前提下尽量减少 RBF 层中的神经元数。其做法则是：从一个节点开始训练，通过检查误差目标使网络自动增加节点。每次循环用使网络产生最大误差所对应的输入矢量产生一个新的 RBF 层节点，然后检查新网络的误差，重复此过程，直到达到目标误差或达到最大神经元数为止。

从结构上看，RBF 网络似乎就是一个具有径向基函数的 BP 网络，它们有着同样的两层网络——隐含层具有径向基函数：一种高斯型指数函数；输出层具有线性激活函数。但是，RBF 网络不是 BP 网络，其原因是：①它不是采用 BP 算法来训练网络权值的；②其训练的算法不是梯度下降法。虽然是两层网络，径向基网络的权值训练是一层一层进行的。在对隐含层中径向基函数权值进行训练时，网络训练的目的是使$w1_{qj}=p_j^q$。由于径向基函数在将其输入放置在原点时输出为 1，而对其他不同的输入值的响应均小于 1，所以设计将每一组输入值p_j^q作为一个径向基函数的原点，而权值$w1_{ij}$代表中心的位置。则通过令$w1_{qj}=p_j^q$使每一个径向基函数只对一组p_j^q响应，从而迅速辨识出p_j^q的大小。然后进行输出层的权值设计。由于输出层是线性函数，网络输出是径向基网络输出的线性组合，从而很容易地达到了从非线性输入空间向输出空间映射的目的。

从功能上看，和 BP 网络一样，RBF 网络可以用来进行函数逼近。并且训练 RBF 网络要比训练 BP 网络所花费的时间要少得多，这是该网络最突出的优点。不过，如前所述，RBF 网络也有自身的缺点。一般而言，即使采用改进的方法来设计，RBF 网络中隐含层节点数比采用 S 型转移函数的前向网络所用的数目要多许多。这是因为 S 型神经元有一个较大范围的输入空间，而 RBF 网络只对输入空间中的一个较小的范围产生响应。结果是，输入空间越大（即输入的数组以及输入的变化范围越大），所需要的 RBF 神经元数越多。

6.3 广义径向基网络

广义径向基函数（General Radial Basis Function，简称 GRBF）网络具有一个径向基层和一个特殊的线性层，如图 6.4 所示。

和普通 RBF 网络一样，在径向基层，神经元的每个激活函数加权输入是输入矢量和各自权值的距离，即$\|W1-P\|$，每个神经元的输入是加权输入与其偏差的点积。

与普通 RBF 网络不同的是，GRBF 网络的 RBF 层输出后，不是立刻进行线性网络的计

算，而是将 RBF 层的输出 a_i 加权求平均值后作为线性函数的输入。RBF 层的输出 a_i 为通过高斯型径向基函数的输出：

$$a_i = \exp\left(-\frac{\|W1_i - P\|^2 \cdot 0.8326^2}{C_i^2}\right) \tag{6.5}$$

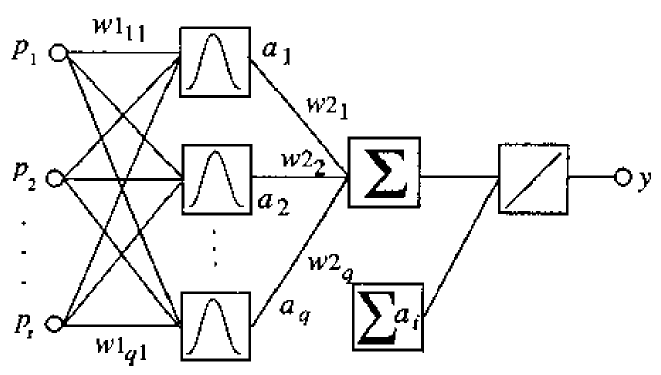


图 6.4 广义径向基网络结构图

整个网络的输出 y 为：

$$y = \frac{\sum a_i \cdot W2_i}{\sum a_i} \tag{6.6}$$

6.4 数字应用对比及性能分析

前向网络中，不论是 BP 网络，还是 RBF 网络，不论它们是用于建模，还是控制，或是其它领域，所利用网络的最基本的功能始终只有一条，那就是函数逼近。本节中给出对函数逼近的应用例子，对两者的性能进行对比分析。

用于曲线逼近的输入/输出数组为：

```
P = -1: 0.1: 1;
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609 ...
      .1336 -.2013 -.4344 -.5000 -.3930 -.1647  .0988 ...
      .3072  .3960  .3449  .1816 -.0312 -.2189 -.3201];
```

输入和输出函数之间的关系图如图 6.5 所示。作为对比，首先对于这 21 组输入/输出对，构造一个隐含层含有 10 个神经元，并采用对数 S 型激活函数，输出层采用线性函数的前向网络，然后分别采用标准的误差反向传播法、改进的带有附加动量法和自适应学习速率相结合的快速算法、Levenberg-Marquardt 算法。而 RBF 网络的训练采用的是自动确定所需要的隐含层节点数的方法。整个设计与训练是在 MATLAB 环境下，用 PII400 计算机进行的，期望误差取 0.01。表 6.1 给出了在不同网络结构以及不同算法下，解决同一问题所花费的时间、循环次数以及浮点操作次数，从中可以看出不论从哪个指标上看，标准

BP 算法与最好的性能指标要差 1~2 个数量级。BP 网络中 L-M 算法最快，它几乎可以和 RBF 网络的指标相当。从整体情况看，RBF 网络显示出快速省时的优点，将 RBF 和 GRBF 网络相比，GRBF 所花费的浮点计算数要明显少于 RBF 网络，这是因为 GRBF 的隐含层神经元数是与训练用的输入组数相等的缘故。不过 RBF 网络通过自动寻优，最终隐含层的神经元数只有 6 个，比 BP 网络的 10 个还要少。实验表明,对于周期变化次数不太多的函数逼近问题，RBF 网络也可以用较少的节点达到较高的逼近精度。

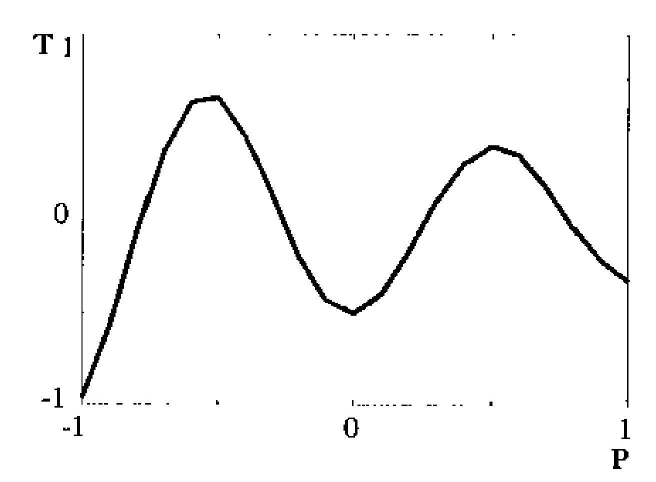


图 6.5 输入和输出函数的关系图

表 6.1 实验对比结果

算法/网络	时间（秒）	循环次数	浮点操作次数
标准 BP 法	14.44	1100	5756585
快速 BP 法	0.82	33	183706
L-M 算法	0.44	2	229663
RBF 网络	0.44	1	32577
GRBF 网络	0.44	1	17741

6.5 小 结

采用 RBF 网络进行函数逼近，与采用 BP 网络相比，前者更容易逼近函数的局部特性，而后者由于是用一个全周函数来逼近函数，因此，对于某些具体问题的收敛速度会慢一些。从实用角度上看，对于函数逼近问题，用 RBF 网络来解决可能更为合适。另外，RBF 网络在功能上与模糊系统有一定的联系，这为我们更好地设计出模糊系统提供了一个新的思路。这一方面的内容将在第 12 章中做进一步的阐述。

第 7 章 模糊理论基础

7.1 引　言

模糊理论是在美国加州大学查德（L. A. Zadeh）教授于 1965 年创立的模糊集合理论的基础上发展起来的，主要包括模糊集合理论、模糊逻辑、模糊推理和模糊控制等方面的内容。

在经典二值逻辑中，假定所有的分类都是有明确边界的，任一被讨论的对象要么属于这一类，要么就不属于这一类，一个命题不是真就是伪，不存在亦真亦伪或者非真非伪的情况。模糊逻辑是对二值逻辑的扩充，它是为解决现实世界中存在的模糊现象而发展起来的，它要考虑被讨论对象属于某一类的程度，一个命题可能亦此亦彼，存在着部分真和部分伪。例如，某一品牌汽车，其中有一部分零件是进口的，其他零件是国产的，要问这种汽车是否是国产的，那该怎么回答呢？在经典逻辑中，要么回答“是”，要么回答“不是”。事实上，我们经常会这样说：“这种车的零件大部分达到了国产化”，在二值逻辑中就无法表达像“大部分”这样不精确的含糊信息；实际上，“大部分”还有一个“程度”的问题。在模糊逻辑中则可利用隶属度来描述“程度”，那么就可说这种车国产化程度是 86%，尚有 14%靠进口。

在处理现实世界的问题方面，越来越多的事实说明模糊逻辑远比二值逻辑更为有效。对那些无法建模的复杂问题，有不少用模糊逻辑能比较好地解决。模糊逻辑之所以十分有用，就在于它能够在经典的二值逻辑不能对变量进行定义的情况下实现有意义而合理的操作。实际上，在我们日常生活中经常不断地会用到模糊逻辑的描述方法，故模糊逻辑是通过模仿人的思维方式来表示和分析不确定、不精确信息的方法和工具。尽管“模糊”这个词在这里容易使人产生误解，实际上在模糊逻辑控制中的每一个特定的输入都对应着一个实际的输出，并且这个输出值是完全可以预测的。所以模糊逻辑本身并不模糊，模糊逻辑并不是“模糊的”逻辑，而是用来对“模糊”进行处理以达到消除模糊的逻辑，它是一种精确解决不精确、不完全信息的方法，其最大特点就是用它可以比较自然地处理人的概念，是一种更人性化的方法。许多事实证明，用逻辑处理和分析现实世界的问题，其结果往往更符合人的要求；加上用模糊逻辑实现控制更能容忍噪音干扰和元器件的变化，使系统适应性更好；模糊逻辑还可使产品开发周期缩短而编程更容易，所以模糊逻辑越来越为更多的科技工作者接受。

模糊逻辑是通过使用模糊集合来工作的。模糊集合与经典集合是不同的。经典集合是具有精确边界的集合。经典集合对集合中的对象关系进行严格划分，一个对象要么是完全属于这个集合，要么就完全不属于这个集合，不存在介于两者之间的情况。例如“包含大于 5 的实数”的经典集合 A 可以表示为：

$$A=\{x \mid x>5\} \tag{7.1}$$

它拥有一个清晰明确的边界 5。如果 x 大于这个数就属于集合 A，否则 x 就不属于集合 A。模糊集合是没有精确边界的集合。这意味着，从“属于一个集合”到“不属于一个集合”

之间的转变是逐渐的，这个平滑的转变是由隶属函数来表征的，例如，如果 X 是论域，且其元素用 x 来定义，X 中的一个模糊集合 A 被定义为：

$$A=\{x,\mu_A(x)|x\in X\} \tag{7.2}$$

其中，$\mu_A(x)$ 被称为 A 中 x 的隶属函数（或 MF），X 的每个元素的隶属函数对应的隶属值在 0～1 之间。

模糊集合具有灵活的隶属关系，它允许在一个集合中部分隶属。对象在模糊集合中的隶属度可以是从 0 到 1 之间的任何值，而不像在经典集合中非得是严格的 0 或 1。这样模糊集合就可以从“不隶属”到“隶属”逐渐地过渡。这样像“快”、“慢”、“热”、“冷”这些本来在经典集合中无法解决的含糊概念就可在模糊集合中得到表达，也就为计算机处理这类带有含糊性的信息提供了一种方法。25℃是暖还是热？用经典集合的概念回答，这要么算暖要么算热；但用模糊术语回答则是“两者都有些，既算暖又算热”。换句话说，用模糊逻辑判断不是一刀切或者黑白分明，而是在两者之间连续渐变。表面看这种含糊是无意义的，但实际上却可通过对这些渐变安排特定的数字，再进行模糊逻辑推理而消除模糊，假如把 25℃作这样的分类，它隶属于暖的程度是 0.6，同时隶属于热的程度是 0.4，然后再用这些数值去得到对问题的精确解。

建立在模糊逻辑基础上的模糊推理是一种近似推理，可以在所获得的模糊信息的前提下进行有效的判断和决策。而基于二值逻辑的理解力推理和归纳推理此时却无能为力，因为它要求前提和结论都是精确的，不能有半点含糊。模糊逻辑推理是不确定性推理方法的一种，其基础是模糊逻辑，它是在二值逻辑三段论的基础上发展起来的，它与传统布尔集合论进行了统一处理，并且用这种推理方法得到的结论与人的思维一致或相近。它是一种以模糊判断为前提，运用模糊语言规则，推出一个新的近似的模糊判断结论的方法。

人们平常如果遇到像“如果 x 小，那么 y 就大”这样的前提，要问“如果 x 很小，y 将怎么样呢”，我们会很自然地想到“如果 x 很小，那么 y 就很大”。人们所使用的这种推理方法就被称作模糊假言推理或似然推理。这是一种近似推理方法。

在 1975 年，查德利用模糊变换关系，提出了模糊逻辑推理的合成规则，建立了统一的数学模型，用于对各种模糊推理作统一处理。模糊假言推理是作为这一合成规则的特殊情况来处理的。

在模糊控制中，所使用的控制规则是人们在实际工作中的经验。这些经验一般是用人们的语言来归纳、描述的。也就是说，模糊控制规则是用模糊语言表示的。通常的模糊控制规则用下面三种条件语言的形式来表示，例如：

①如果水温偏高，那么就加一些冷水；

②如果衣服很脏，那么洗涤时间应很长，否则洗涤时间不必太长；

③如果温度偏高并且不断上升，那么应加大压缩机的制冷量。

为了形式化和数学处理上的方便，上述条件语句也可分别表示为：

①如果 x 是 A，那么 y 是 B；

②如果 x 是 A，那么 y 是 B，否则 y 是 C；

③如果 x 是 A 并且 y 是 B，那么 z 是 C。

模糊推理系统是建立在模糊集合、模糊规则和模糊推理等概念基础上的先进的计算系统。它在诸如系统建模、自动控制、数据分类、决策分析、专家系统、时间序列预测、机

器人控制和模式识别等众多领域中得到了成功的应用。由于其多学科的自然属性，模糊推理系统被称为许多不同的名字，如基于模糊规则系统、模糊专家系统、模糊模型、模糊联想记忆、模糊逻辑控制器，或简单地称之为模糊系统。

7.2 模糊集合及其隶属函数

7.2.1 模糊集合的定义

在经典集合论中，任何一个元素与任何一个集合之间的关系，只有“属于”和“不属于”两种情况，两者必居其一，而且只居其一，绝对不允许模棱两可。比如“不大于5的自然数”是一个清晰的概念，该概念的内涵和外延均是明确的。可是，我们也经常遇到没有明确外延的概念，这种概念实质上是模糊概念。例如，“比 5 大得多的自然数”就是一个模糊概念。可以想象，无法划定一个明确的界限，使得在这个界限内所有自然数都比 5 大得多，而界限外的所有自然数都不比 5 大得多。只能说某个数属于“比 5 大得多”的程度高，而另一个数属于“比 5 大得多”的程度低，比如 50 属于“比 5 大得多”的程度比 10 属于“比 5 大得多”的程度高。

查德在 1965 年把经典集合中的元素对集合的隶属度只能取 0 和 1 这两个值，推广到可以取区间[0, 1]中的任意一个数值。即可以用隶属度定量去描述论域 U 中的元素符合概念的程度，实现了对经典集合中绝对隶属关系的扩充，从而用隶属函数表示模糊集合，用模糊集合表示模糊概念。

下面是对模糊集合的一般定义。

设 U 为一可能是离散或连续的集合，U 被称为论域(Universe of Discourse)，用$\{u\}$表示论域 U 的元素。模糊集合是用隶属函数来表示的。

定义 1 论域 U 中的模糊子集 A，是以隶属函数 μ_A 为表征的集合，即由映射：

$$\mu_A:\ U \to [0,1] \tag{7.3}$$

确定论域 U 的一个模糊子集 A。μ_A 称为模糊子集的隶属函数，$\mu_A(u)$称为 u 对 A 的隶属度，它表示论域 U 中的元素 u 属于其模糊子集 A 的程度。它在[0, 1]闭区间内可连续取值，隶属度也可简记为 $A(u)$。

关于模糊子集 A 和隶属函数 μ_A，做如下几点说明：

①论域 U 中的元素是分明的，即 U 本身是普通集合，只是 U 的子集是模糊集合，故称 A 为 U 的模糊子集，简称模糊集。

②$\mu_A(u)$是用来说明 u 隶属于 U 的程度的。$\mu_A(u)$的值越接近 1，表示 u 从属于 A 的程度越大；反之，$\mu_A(u)$的值越接近于 0，则表示 u 从属于 A 的程度越小。显然，当$\mu_A(u)$的值域为 $\{0,1\}$ 时，隶属函数 μ_A 已蜕变为经典集合的特征函数，模糊集合 A 也就蜕变成为一个经典集合。因此，可以这样来概括经典集合和模糊集合间的互变关系，即模糊集合是经典集合在概念上的拓广，或者说经典集合是模糊集合的一种特殊形式；而隶属函数则是特征函数的扩展，或者说，特征函数只是隶属函数的一个特例。

③模糊集合完全由它的隶属函数来刻画。隶属函数是模糊数学的最基本概念，借助于它才能对模糊集合进行量化。正确地建立隶属函数，是使模糊集合能够恰当地表达模糊集合的关键，是利用精确的数学方法去分析处理模糊信息的基础。

下面以人对室温（0℃ ~ 40℃）的感觉为例，来看看如何用模糊集合表示人对事物和现象形成的概念。在一般情况下，大部分人都把 15℃～28℃的室温称作“舒适”的温度，而把 15℃以下称为“凉”，28℃以上称为“热”。用经典集合来定义，如图 7.1 所示，小于 15℃的温度，哪怕是 14.9℃，也只能属于“凉”的温度，14.9℃与 15℃只相差 0.1℃，就把 15℃归为“舒适”，而把 14.9℃归为“凉”，这合理吗？显然就人的感觉而言，这是不恰当的。若用模糊集合来定义，就要用对某一个模糊元素具有 0 到 1 之间连续变化隶属度的特征函数来描述，模糊集合的特征函数就称作隶属函数（Membership Function），图 7.2 就是一种表示方法。在模糊逻辑中，与人的感觉一致，小的温度变化只会引起系统性能的逐渐变化，14.9℃与 15℃属于同一个集合的程度是很接近的。这种情况下，32℃被认为属于“舒适”的程度是 0.3，还同时属于“热”的程度是 0.7。由此可见，在模糊逻辑系统中，温度的较小变化，在系统执行中将导致一个比较合乎情理的变化。

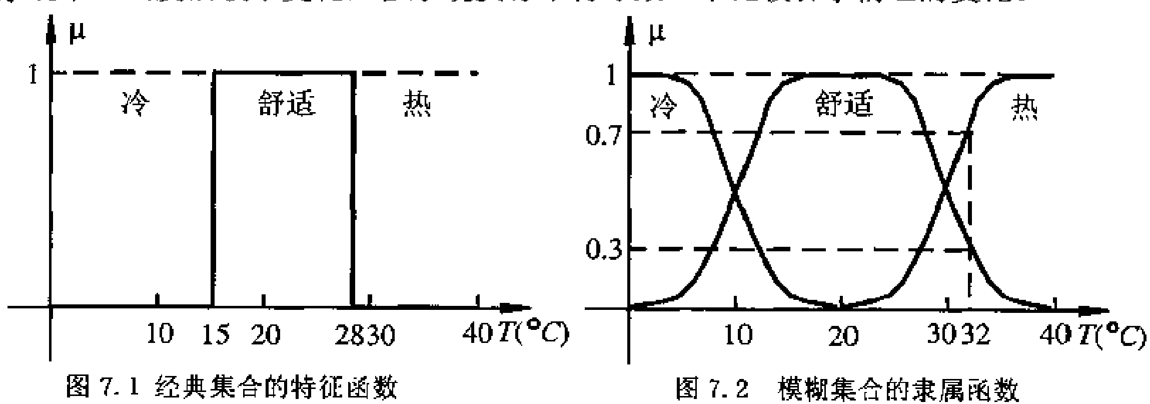


图 7.1 经典集合的特征函数　　图 7.2 模糊集合的隶属函数

7.2.2 模糊集合的表示方法

就论域的类型而言，模糊集合有下列两种表示方法：

①设论域 U 是有限域，即 $U=\{u_1,u_2,\ldots,u_n\}$，U 上的任意一个模糊集合 A，其隶属函数为 $\mu_A(u_i), i=1,2,\ldots,n$，则此时 A 可表示成：

$$\sum_{i=1}^{5}\mu_{\mathrm{A}}(u_i)/u_i \tag{7.4}$$

这里的 $\sum$ 并不表示“求和”，$\mu_{\mathrm{A}}(u_i)/u_i$ 也不是分数，只是借用来表示集合的一种方法，它们只有符号意义，表示 A 对模糊集合的隶属程度是 $\mu_A(u_i)$ 。

例 7.1 设室温的论域 $U=\{0℃, 10℃, 20℃, 30℃, 40℃\}$，模糊集合 A 表示“舒适的温度”，则 A 可以定义为：

$$A=\text{“舒适的温度”}=\sum_{i=1}^{5}\mu_{\mathrm{A}}(u_i)/u_i=0.25/0+0.5/10+1.0/20+0.5/30+0.25/40$$

其中，分式的含义是隶属度 / 温度值。

②设论域 U 是无限集合，此时 U 上的一个模糊集合 A 可表示成

$$A = \int_U \mu_A(u) / u \tag{7.5}$$

注意，同样，这里的$\int$ 不再表示“积分”，只代表一种记号；$\mu_A(u)/u$ 的意义则和有限情况是一致的。

7.2.3 模糊集合的并、交、补运算

模糊集合的运算种类很多，但最常用的还要数模糊集合的并集、交集和补集运算。模糊集合由隶属函数定义而成，其运算也可以由隶属函数来定义。设 A、B 为 U 中两个模糊集合，隶属函数分别为 μ_A 和 μ_B，则模糊集合理论中的并、交、补等运算可通过它们的隶属函数来定义。

定义 2 A 与 B 的并集记作 $A \cup B$，其隶属函数 $\mu_{A\cup B}$ 对所有 $u \in U$ 被逐点定义为取大运算，即

$$\mu_{A\cup B}(u) = \max\{\mu_A(u), \mu_B(u)\} = \mu_A(u) \vee \mu_B(u) \tag{7.6}$$

定义 3 A 与 B 的交集记作 $A \cap B$，其隶属函数 $\mu_{A\cap B}$ 对所有 $u \in U$ 被逐点定义为取小运算，即

$$\mu_{A\cap B}(u) = \min\{\mu_A(u), \mu_B(u)\} = \mu_A(u) \wedge \mu_B(u) \tag{7.7}$$

定义 4 A 的补集记作 $\overline{A}$，其隶属函数 $\mu_{\overline{A}}$ 对所有 $u \in U$ 被逐点定义为

$$\mu_{\overline{A}}(u) = 1 - \mu_A(u) \tag{7.8}$$

这里的 $\vee$、$\wedge$ 符号称为查德算子，为模糊逻辑中的运算符号，在无限集合中，它们分别表示 sup 和 inf，在有限元素之间则表示 max 和 min，即取最大值和最小值。

7.2.4 模糊集合的隶属函数

查德教授在 1965 年发表的论文《模糊集合》（Fuzzy Sets）中首次提出了表达事物模糊性的重要概念——隶属函数，借助于隶属函数可以表达一个模糊概念从“完全不属于”到“完全属于”的过渡，能够对所有的模糊概念进行定量表示。隶属函数的提出奠定了模糊理论的数学基础。这样，像“冷”和“热”这些在经典集合中无法解决的模糊概念就可在模糊集合中得到有效表达，这就为计算机处理这种语言信息提供了一种可行的方法。

在经典集合中，特征函数只能取 0 和 1 两个值，即特征函数与{0, 1}相对应；而在模糊集合中，其特征函数的取值范围从两个元素的集合扩大到在[0, 1]区间连续取值。为了把两者区分开来，就把模糊集合的特征函数称为隶属函数。若隶属函数的取值只取 0 和 1，那么模糊集合就缩简成经典集合。从这个意义上说，模糊集合的隶属函数是经典集合特征函数的扩展和一般化。模糊集合是通过隶属函数来定义的，准确地确定隶属函数是运用模糊集合理论解决实际问题的基础。隶属函数的确定实质上是人们对客观事物中介过渡的定性描述，这种描述本质上是客观的。但因为每个人对同一模糊概念的认识和理解上存在差异，因此又含有一定的主观因素。对于同一个模糊概念，不同的人会建立不完全相同的隶

属函数。所以从理论上说，即使根据专家的经验确定的隶属函数，这种没有理论化的方法也不能保证其正确性，因为任何人的经验和知识都是有局限性的。在这个方面，国内外学者已经进行了大量的研究，提出了各种各样的确定方法，诸如模糊统计法、函数分段法、二元对比排序法、对比平均法、滤波函数法、示范法和专家经验法等等方法。不过，在实际应用中，虽然用不同方法确定了不同的模糊集合的隶属函数，在一定范围内，尽管实现控制过程的细节和达到目标的过程细节和响应时间可能有差别，但模糊逻辑控制却都能实现控制，达到预期目标。换句话说，隶属函数的确定并不是唯一的，允许有不同的组合。所以人们为了简化计算，很多模糊逻辑控制的隶属函数曲线都是取三角形。实际上，根据模糊统计方法得到的隶属函数通常都是钟形的，所以三角形隶属函数并不是最佳函数，只是一种近似。计算机模拟实验发现，实际上，隶属函数的形状会很微妙地影响着整个模糊系统的过程，例如会影响单片机实现模糊化、解模糊化的时间和对查询表存储空间的要求。现在普遍采用三角形、梯形和单值线形状（又称棒形），是因为实践证明它能满足一般要求，又可简化计算，故被广泛采用。

如果按定义，模糊集合的隶属函数可取无穷多个值，这在实际使用中是难以确定的，所以一般可进行如下简化：把最大适合区间的隶属度定为 1.0，中等适合区间的隶属度定为 0.5，小适合区间的隶属度定为 0.25，最小隶属度（即不隶属）为 0.0。再对一些常用的基本隶属图形进行定义。基本的隶属函数图形可分成三类：左大右小的偏小型下降函数（通常称作 Z 函数）、对称型凸函数（通常称作 Π 函数）和右大左小的偏大型上升函数（通常称作 S 函数）。这三种隶属函数的形状如图 7.3 所示。

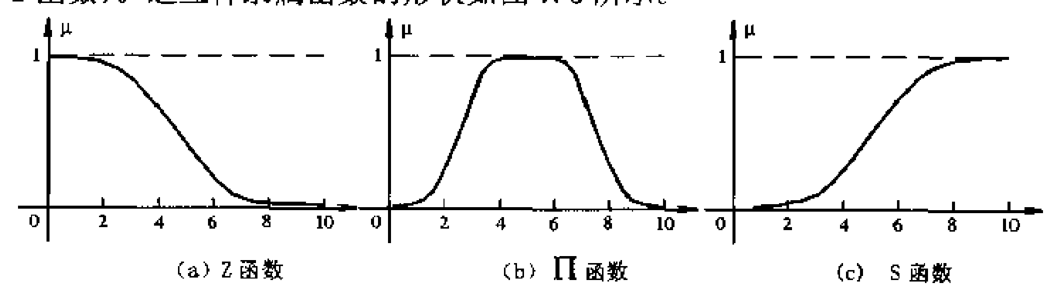


图 7.3 基本的隶属函数图形

最简单的隶属函数是三角形，它是用直线形成的；梯形隶属函数实际上是由三角形截顶所得。这两种直线形隶属函数都具有简单的优势，因而经常被人们使用。它们的形状见图 7.4（a）和图 7.4（b）。

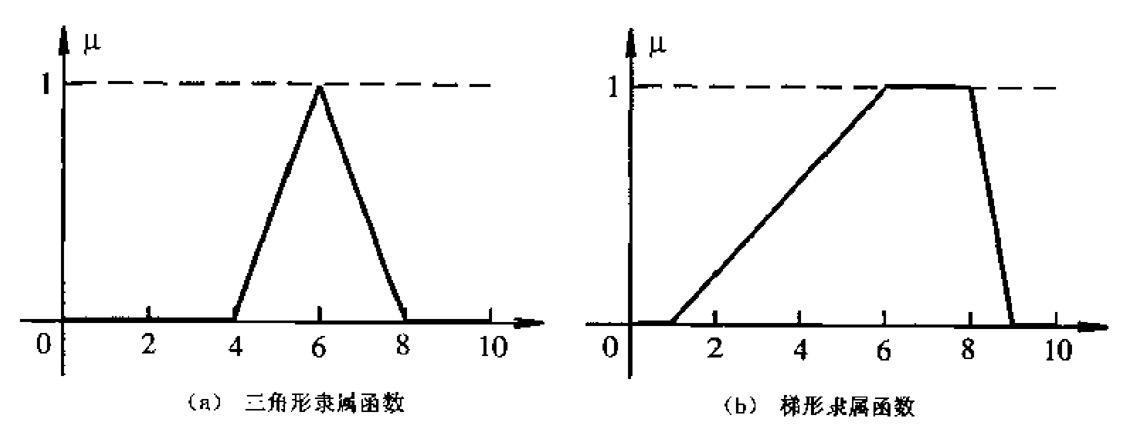


图 7.4 直线型隶属函数

7.3 模糊逻辑

7.3.1 二值逻辑、多值逻辑和模糊逻辑

研究思维形式和规律的学科叫逻辑学。用经典逻辑表达命题的形式如“明天将会下雨是真”，其反面则是“明天将不会下雨是真”。二值逻辑不承认有任何过渡。波兰逻辑学家和哲学家卢卡瑟维兹（J. Lukasiewicz）将二值逻辑扩展成三值逻辑，即加上了另外一种表述“明天将下雨是可能的”，这种表述的逻辑值是 1/2，他用 1 表示真，0 表示假，另外用 1/2 表示可能性，这看起来好像仅仅是插入了一个值，然而却是一个突破，因为由此而产生了多值逻辑。

多值逻辑否定了逻辑真值的绝对两极性，认为逻辑真值具有离散的中间过渡，似乎在某种程度上具有亦此亦彼性。但是多值逻辑是通过穷举中介的方式表现这种过渡性，把所有中介看成是若干完全分立离散、界限分明的对象，而不承认相邻中介是相互渗透、交叉重叠的。因此，多值逻辑本质上仍然属于精确逻辑，而不是真正的亦此亦彼的逻辑。

模糊逻辑是在卢卡斯维兹多值逻辑基础上发展起来的，如前所述，多值逻辑中命题的真值可取从 0 到 1 之间的任何值，但此值是确切的。然而在许多情况下，要给命题的真实程度赋予确切数值也是困难的。但是人们用某些约定的模糊语言却能对模糊命题给予贴切的描述，这些语言虽然不是精确的，然而相互之间却都能理解接受，并且一般不但不会引起误解，反而显得更贴切有效，体现了人脑模糊思维的逻辑特征。这说明这些模糊语言具有逻辑真值的功能。由此用带有模糊限定算子（例如，很、略、比较、非常等）的从自然语言中提炼出来的语言真值（如年轻、非常年轻等）或者模糊数(例如，大约 25、45 左右等）来代替多值逻辑中命题的确切数字真值，就构成模糊语言逻辑，通常就简称为模糊逻辑。在经典二值逻辑中实际上也有语言真值，那就是真和假，但是在模糊逻辑中则有无穷多个语言真值。

7.3.2 模糊逻辑的基本运算

模糊逻辑的基本运算有以下 5 种：

①模糊逻辑“补”： $\overline{P} = 1 - P$

②模糊逻辑“取小”： $P \wedge Q = \min(P, Q)$ （取两个真值中小的一个）（对应于二值逻辑中的“与”）

③模糊逻辑“取大”： $P \vee Q = \max(P, Q)$ （取两个真值中大的一个）（对应于二值逻辑中的“或”）

④模糊逻辑“蕴含”： $P \to Q = ((1 - P) \vee Q) \wedge 1$

⑤模糊逻辑“等价”： $P \leftrightarrow Q = (P \to Q) \wedge (Q \to P)$

除这 5 种运算外，为模糊逻辑运算又定义了 3 种限界运算。

对各个元素而言，分别相加，相加后的值比 1 小的作为限界和，而把大于 1 的部分作为限界积。

⑥模糊逻辑限界积：　$P \otimes Q = (P + Q - 1) \vee 0 = \max((P + Q - 1, 0)$

⑦模糊逻辑限界和：　$P \oplus Q = (P + Q) \wedge 1 = \min((P + Q, 1)$

⑧模糊逻辑限界差：　$P - Q = (P - Q) \vee 0$

因为二值逻辑真值只能取 0 或者 1，它除了可以用解析法进行处理化简外，还有比较直观的真值表图解形式的卡诺图方法。模糊逻辑公式常称为模糊逻辑函数。由于模糊逻辑函数可在从 0 到 1 区间取任意值，所以模糊逻辑公式就不能用传统的卡诺图方法，而只能用解析法 Veitch 图的推广——模糊图来处理，这在实际应用中会带来困难。所以为了简化处理，又常把模糊函数变量分成若干有限段，这样就可用多值逻辑的方法来处理模糊逻辑的问题。

根据以上模糊逻辑函数运算的定义，可以推导出模糊逻辑函数的下列基本公式：

①幂等律：　$P \vee P = P$

　　　　　$P \wedge P = P$

②交换律：　$P \vee Q = Q \vee P$

　　　　　$P \wedge Q = Q \wedge P$

③结合律：　$P \vee (Q \vee R) = (P \vee Q) \vee R$

　　　　　$P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$

④吸收律：　$P \vee (P \wedge Q) = P$

　　　　　$P \wedge (P \vee Q) = P$

⑤分配律：　$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$　　(7.9)

　　　　　$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$

⑥双否律：　$\overline{\overline{P}} = P$

⑦德•摩根律：　$\overline{P \vee Q} = \overline{P} \wedge \overline{Q}$

　　　　　$\overline{P \wedge Q} = \overline{P} \vee \overline{Q}$

⑧常数运算法则：　$1 \vee P = 1$　　$0 \vee P = P$

　　　　　$1 \wedge P = 0$　　$1 \wedge P = 0$

这里特别要注意的是，在二值逻辑中有互补律：$P \vee \overline{P} = 1$；$P \wedge \overline{P} = 0$。但是在模糊逻辑中没有互补律，因为

$$P \vee \overline{P} = \max(P, 1 - P)$$

$$P \wedge \overline{P} = \min(P, 1 - P) \tag{7.10}$$

并且一般说来前者不等于 1，后者不等于 0。

利用这些基本公式就可化简模糊逻辑函数，以便根据化简得到的结果来组成最简模糊逻辑电路。这具有非常重要的现实意义。

7.3.3 模糊关系和模糊矩阵

描述元素之间是否相关的数学模型称为关系，描述模糊元素之间相关程度的数学模型称为模糊关系。为了区别于模糊关系，又称关系为普通关系。显然，模糊关系是普通关系的拓展和发展，而普通关系可视为模糊关系的特例。模糊关系是模糊数学的重要组成部分。当论域有限时，可用模糊矩阵表示模糊关系。模糊矩阵为模糊关系的运算带来了极大的方便，成为模糊关系的主要运算工具。

下面是关于模糊关系的定义。

定义 5 集合 U 与 V 之间的直积

$$U \times V = \{(u,v) | u \in U, v \in V\} \tag{7.11}$$

中的一个模糊子集 R 被称为 U 到 V 的模糊关系，又称为二元模糊关系，其特性可以由下面的隶属函数来描述：

$$\mu_R : U \times V \to [0,1] \tag{7.12}$$

在论域 $U=V$ 时，称 R 为 U 上的模糊关系，当论域为 n 个集合 U_i（$i=1, 2, \ldots, n$）的直积 $U_1 \times U_2 \times \cdots \times U_n$ 时，它们所对应的模糊关系 R 则称为 n 元模糊关系。

不同乘积空间上的模糊关系可以通过复合运算结合在一起。模糊关系复合运算已经提出了若干种，最典型的是由查德所提出的最大—最小复合运算，其定义如下：

定义 6 设 R 和 S 分别为 $U \times V$ 和 $V \times W$ 上的模糊关系。所谓 R 和 S 的合成，是指下列定义在 $U \times W$ 上的模糊关系，记作 $R \circ S$：

$$\begin{aligned} R \circ S \longleftrightarrow \mu_{R \circ S}(u,w) &= \max \min[\mu_R(u,v), \mu_S(v,w)] \\ &= \vee_v \{\mu_R(u,v) \wedge \mu_S(v,w)\} \end{aligned} \tag{7.13}$$

这里 $\wedge$ 代表取小（min），$\vee$ 代表取大(max)。

尽管最大—最小复合运算得到了广泛的应用，但是很难对其进行数学分析。为了增强数学分析及实现的能力，在模糊关系的合成中，常用乘法代替最大—最小复合运算中的取小运算，获得最大—乘积（max-product）复合运算，即：

$$R \circ S \longleftrightarrow \mu_{R \circ S}(u,w) = \vee_v \{\mu_R(u,v) \bullet \mu_S(v,w)\} \tag{7.14}$$

因为 U 和 V 之间的模糊关系是定义在 $U \times V$ 上的模糊子集，因此，模糊集合之间的运算能够直接利用模糊关系运算。例如，设 R 和 S 是 $U \times V$ 上的模糊关系，则有：

$$\begin{aligned}
&\text{①并集} \quad R \cup S \longleftrightarrow \mu_{R \cup S}(u,v) = \mu_R(u,v) \vee \mu_S(u,v) \\
&\text{②交集} \quad R \cap S \longleftrightarrow \mu_{R \cap S}(u,v) = \mu_R(u,v) \wedge \mu_S(u,v) \\
&\text{③补集} \quad \overline{R} \longleftrightarrow \mu_{\overline{R}}(u,v) = 1 - \mu_R(u,v) \\
&\text{④等价关系} \quad R = S \longleftrightarrow \mu_R(u,v) = \mu_S(u,v), \forall u \in U, v \in V \\
&\text{⑤包含关系} \quad R \subseteq S \longleftrightarrow \mu_R(u,v) \le \mu_S(u,v), \forall u \in U, v \in V
\end{aligned} \tag{7.15}$$

模糊关系通常可以用模糊矩阵、模糊图和模糊集合表示法等形式来表示。通常用模糊矩阵来表示二元模糊关系。

模糊矩阵的定义如下：

当 $U=\{u_i \mid i=1,2,\cdots,m\}$ 和 $V=\{v_i \mid i=1,2,\cdots,n\}$ 是有限集合时，$U\times V$ 的模糊关系 R 可用下列 $m\times n$ 阶矩阵来表示

$$R=\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix} \tag{7.16}$$

上式中，元素 $r_{ij}=\mu_R(u_i,v_j)$ 。由此表示模糊关系的矩阵，被称为模糊矩阵。由于 μ_R 的取值区间为[0, 1]，因此模糊矩阵元素 r_{ij} 的值也在[0, 1] 区间。显然，模糊矩阵是普通矩阵的特例。当 $m = n$ 时，称 R 为 n 阶模糊方阵；当 r_{ij} 全为 0 时，称 R 为零矩阵，记为 0；当 r_{ij} 全为 1 时，称 R 为全矩阵，记为 E；当 r_{ij} 只在{0, 1}中取值时，称 R 为布尔矩阵，它对应一个普通关系。

由于模糊矩阵本身是表示一个模糊关系子集，因此根据模糊集的交、并、补运算定义，模糊矩阵也可作相应的运算。

例 7.2 最大—最小和最大—乘积的复合运算。

设 R = "u 与 v 相关"，S = "v 与 w 相关" 是分别定义在 $U\times V$ 和 $V\times W$ 上的两个模糊关系，其中 $U=\{1,2,3\}$，$V=\{\alpha,\beta,\gamma,\delta\}$，$W=\{a,b\}$，假设 R, S 以如下的关系矩阵来表示：

$$R=\begin{array}{c} \begin{matrix}\alpha & \beta & \gamma & \delta\end{matrix} \\ \begin{bmatrix} 0.1 & 0.3 & 0.5 & 0.7 \\ 0.4 & 0.2 & 0.8 & 0.9 \\ 0.6 & 0.8 & 0.3 & 0.2 \end{bmatrix} \end{array}\begin{matrix}1\\2\\3\end{matrix}, \qquad S=\begin{array}{c} \begin{matrix}a & b\end{matrix} \\ \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.3 \\ 0.5 & 0.6 \\ 0.7 & 0.2 \end{bmatrix} \end{array}\begin{matrix}\alpha\\\beta\\\gamma\\\delta\end{matrix}$$

现在计算 $R\circ S$，它的含义是基于 R 和 S 导出的模糊关系 "u 与 w 相关"。

如果采用最大—最小复合运算的合成方法，可得

$$\mu_{R\circ S}=\begin{bmatrix} 0.1 & 0.3 & 0.5 & 0.7 \\ 0.4 & 0.2 & 0.8 & 0.9 \\ 0.6 & 0.8 & 0.3 & 0.2 \end{bmatrix}\circ\begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.3 \\ 0.5 & 0.6 \\ 0.7 & 0.2 \end{bmatrix}$$

$$=\begin{bmatrix} (0.1\wedge0.9)\vee(0.3\wedge0.2)\vee(0.5\wedge0.5)\vee(0.7\wedge0.7) & (0.1\wedge0.1)\vee(0.3\wedge0.3)\vee(0.5\wedge0.6)\vee(0.7\wedge0.2) \\ (0.4\wedge0.9)\vee(0.2\wedge0.2)\vee(0.8\wedge0.5)\vee(0.9\wedge0.7) & (0.4\wedge0.1)\vee(0.2\wedge0.3)\vee(0.8\wedge0.6)\vee(0.9\wedge0.2) \\ (0.6\wedge0.9)\vee(0.8\wedge0.2)\vee(0.3\wedge0.5)\vee(0.2\wedge0.7) & (0.6\wedge0.1)\vee(0.8\wedge0.3)\vee(0.3\wedge0.6)\vee(0.2\wedge0.2) \end{bmatrix}$$

$$=\begin{array}{c} \begin{matrix}\mathrm{a} & \mathrm{b}\end{matrix} \\ \begin{bmatrix} 0.7 & 0.5 \\ 0.7 & 0.6 \\ 0.6 & 0.3 \end{bmatrix} \end{array}\begin{matrix}1\\2\\3\end{matrix}$$

另外，如果采用最大—乘积合成法，则有：

$$\mu_{R\circ S}=\begin{bmatrix}0.1 & 0.3 & 0.5 & 0.7\\0.4 & 0.2 & 0.8 & 0.9\\0.6 & 0.8 & 0.3 & 0.2\end{bmatrix}\circ\begin{bmatrix}0.9 & 0.1\\0.2 & 0.3\\0.5 & 0.6\\0.7 & 0.2\end{bmatrix}$$

$$=\begin{bmatrix}(0.1*0.9)\vee(0.3*0.2)\vee(0.5*0.5)\vee(0.7*0.7) & (0.1*0.1)\vee(0.3*0.3)\vee(0.5*0.6)\vee(0.7*0.2)\\(0.4*0.9)\vee(0.2*0.2)\vee(0.8*0.5)\vee(0.9*0.7) & (0.4*0.1)\vee(0.2*0.3)\vee(0.8*0.6)\vee(0.9*0.2)\\(0.6*0.9)\vee(0.8*0.2)\vee(0.3*0.5)\vee(0.2*0.7) & (0.6*0.1)\vee(0.8*0.3)\vee(0.3*0.6)\vee(0.2*0.2)\end{bmatrix}$$

$$=\begin{bmatrix}0.09\vee0.06\vee0.25\vee0.49 & 0.01\vee0.09\vee0.30\vee0.14\\0.36\vee0.04\vee0.40\vee0.63 & 0.04\vee0.06\vee0.48\vee0.18\\0.54\vee0.16\vee0.15\vee0.14 & 0.06\vee0.24\vee0.18\vee0.04\end{bmatrix}$$

$$\begin{matrix} & a & b & \\ = & \left[\begin{matrix}0.49\\0.63\\0.54\end{matrix}\right. & \left.\begin{matrix}0.30\\0.48\\0.24\end{matrix}\right] & \begin{matrix}1\\2\\3\end{matrix}\end{matrix}$$

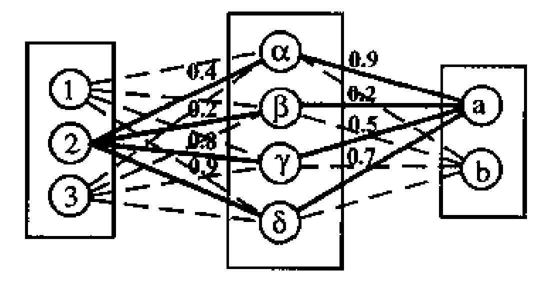


图 7.5 模糊关系的图形表示

图 7.5 所示为两个模糊关系的合成，其中 U 中元素 2 和 W 中元素 a 之间的关系，由连接这 2 个元素的 4 条可能路径（实线）表示。2 和 a 之间的相关度为这 4 条路径强度的最大值，而每条路径的强度等于各元素连接强度的极小值（或乘积）。

7.3.4 模糊语言及其算子

模糊逻辑原则上是一种模拟人类思维的逻辑。人在日常生活中相互交流信息时，用的是自然语言。自然语言是指人们在日常生活和工作中所使用的语言，实际上是以字或词为符号的一种符号系统。自然语言的奇妙作用就是用带有模糊性的词句，对客观现象和事物作出概括性反应，使得人可以用最少的言辞传递最大的信息量。自然语言可以对连续性变化的现象和事物进行概括抽象并作模糊分类，例如，早晨、上午、中午、下午和傍晚、晚上和夜里等，另外这里有个约定俗成的“常识”在起作用，所用的言辞直接与现实世界相对应。这就是说，自然语言具有灵活性。

人类使用的词语是如此丰富的资源，使人难以割舍。在模糊逻辑中，采用词语和模糊语言。在实际中，通常首先要对问题进行初始化，随后对其进行参数调整，在这一过程中，词语起着较大的作用。从这个角度来看，模糊语言是用来达到计算目的的一种手段。进行计算就是将输入转化成输出，或将原因转化成结果，或将问题转化成答案，任何与自然语言的匹配均只是手段而不是目的。目的是设计系统和解决问题。

要想用机器来模仿人的思维、推理和判断，也必须引入语言变量。查德教授在 1975

年提出了语言变量的概念，语言变量实际上是一种模糊变量，它用词句而不是用数字来表示变量的“值”。

通常，把含有模糊概念的语言称为模糊语言。查德首先从语义角度对自然语言进行集合描述，给出了一个集合描述的语言系统。模糊语言也具有它自己的组成要素和语法规则。

1）单词

单词是语言构成的基本要素，也是表达概念的最小单位，因此也称为原子单词。它是不可分割的，如“天”、“地”、“人”、“日”、“月”等。对于一个给定的论域 U，与 U 相关的一类单词就构成了一个集合 A。语义是通过 A 到 U 的对应关系 R 来表达的，R 通常是一个模糊关系。对于任意一个固定的单词 $\alpha\in A$，记作：

$$R(\alpha,u)=\mu_A(u) \tag{7.17}$$

上式表示的是论域 U 上的一个模糊子集，并用大写字母 A 表示与 α 对应的关系。若设：

$$\mu_R:A\times U\to[0,1]$$

其隶属函数 $\mu_A(\alpha,u)$ 表示属于集合 A 的单词 α 和属于论域 U 上元素 u 之间关系的程度。

例如，设论域 U 为夏天气温，单词 α 为“高温”，元素 u 为“日最高气温”，则有：

$$\mu_A(\alpha,35℃)=0.2;\quad \mu_A(\alpha,36℃)=0.4;\quad \mu_A(\alpha,37℃)=0.6;$$

$$\mu_A(\alpha,38℃)=0.8;\quad \mu_A(\alpha,39℃)=0.9;\quad \mu_A(\alpha,40℃)=1.0。$$

这里，模糊子集 A 为：

$$A=\frac{0.2}{35}+\frac{0.4}{36}+\frac{0.6}{37}+\frac{0.8}{38}+\frac{0.9}{39}+\frac{1.0}{40}$$

由于“高温”本身是一个模糊性单词，因此，α 称为“模糊的”。如果论域 U 和元素 u 均不变，改变单词 α 为“高于 3 7℃气温”，则：

$$\mu_A(\alpha,35℃)=0;\quad \mu_A(\alpha,36℃)=0;\quad \mu_A(\alpha,37℃)=0;$$

$$\mu_A(\alpha,38℃)=1;\quad \mu_A(\alpha,39℃)=1;\quad \mu_A(\alpha,40℃)=1。$$

此时，模糊集合 A 退化为一个清晰集合，即：

$$A=\{38℃，39℃，40℃\}$$

而这里的“高于 37℃气温”是一个清晰的概念，因此 α 是“清晰的”。

2）词组

由连接词“或”、“且”将两个或多个单词相连接，或者在单词前面加“非”，即可构成词组。这些连接词在逻辑上分别对应于集合运算的 $\cup$、$\cap$ 和非。例如：

人＝男人或女人＝[男人] $\cup$ [女人]

非机动车＝$\overline{[机动]}\cap$[车子]

由上例可知，单词可以组合成词组，词组可以分解为单词，它们可以统称为“词”。

3）语言算子

为了对模糊的自然语言形式化和定量化，进一步区分和刻画模糊值的程度，常常还借用自然语言中的修饰词，诸如“非常”、“比较”、“极其”，还有“稍微”、“相当”、“大约”、“近似、“倾向于”来描述模糊值。为此引入语言算子的概念。语言算子通常又可分为三类：语气算子、模糊化算子和判定化算子。

（1）语气算子

语气算子用于表达模糊值的肯定程度。这又可分为两种情况：一种是有强化作用的语气算子，例如“很”、“极”等，可以使模糊值的隶属度的分布向中央集中，常称为集中化算子，集中化算子在图形上有使模糊值尖锐化的倾向；另一种是有弱化作用的语气算子，例如“较小”、“稍微”等，可以使模糊值的隶属度的分布由中央向两边弥散，常称为松散化算子。松散化算子在图形上有使模糊值平坦的倾向。

为了规范语气算子的意义，查德曾对此作了如下约定：用 H_λ 作为语气算子来定量描述模糊值。若模糊值为 A，则把 H_λ 定义成

$$H_\lambda = A^\lambda \tag{7.18}$$

设 H_4 代表“极”，或者“非常非常”，其意义是对描述的模糊值求 4 次方；
设 H_2 代表 “很”或者“非常”，其意义是对描述的模糊值求 2 次方；
设 $H_{1/2}$ 代表“较”或者“相当”，其意义是对描述的模糊值求 1/2 次方；
设 $H_{1/4}$ 代表“稍”或者“略微”，其意义是对描述的模糊值求 1/4 次方；

例如，论域 $U=[0, 200]$，而 O 表示单词“年老”，那么（$H_\lambda O$）随着 λ 取不同的值，就可以表示出“年老”的程度。

当 $\lambda>1$ 时，不妨设 $H_{1.25}$ 为“相当”，H_2 为“很”，H_4 为“极”，则

$$[\text{相当老}](u)=\begin{cases}0 & 0\le u\le 50\\ \left[1+\left(\dfrac{u-50}{5}\right)^{-2}\right]^{-1.25} & 50<u\le 200\end{cases}$$

$$[\text{很老}](u)=\begin{cases}0 & 0\le u\le 50\\ \left[1+\left(\dfrac{u-50}{5}\right)^{-2}\right]^{-2} & 50<u\le 200\end{cases}$$

$$[\text{极老}](u)=\begin{cases}0 & 0\le u\le 50\\ \left[1+\left(\dfrac{u-50}{5}\right)^{-2}\right]^{-4} & 50<u\le 200\end{cases}$$

当 $\lambda<1$ 时，不妨设 $H_{0.25}$ 为“微”，$H_{0.5}$ 为“略”，$H_{0.75}$ 为“比较”，则

$$[\text{微老}](u)=\begin{cases}0 & 0\le u\le 50\\ \left[1+\left(\dfrac{u-50}{5}\right)^{-2}\right]^{-0.25} & 50<u\le 200\end{cases}$$

$$
[\text{略老}](u)=\begin{cases}0 & 0\le u\le 50\\ \left[1+\left(\dfrac{u-50}{5}\right)^{-2}\right]^{-0.5} & 50<u\le 200\end{cases}
$$

$$
[\text{比较老}](u)=\begin{cases}0 & 0\le u\le 50\\ \left[1+\left(\dfrac{u-50}{5}\right)^{-2}\right]^{-0.75} & 50<u\le 200\end{cases}
$$

注意，语气算子是在针对隶属函数为指数型函数时得出的。若采用其他如三角形或梯形隶属函数，此算子不起作用。

由于隶属函数的取值范围在闭区间[0, 1]，由于集中化算子的幂乘运算的次数大于1，乘方运算后变小，即隶属函数曲线趋于尖锐化，而且幂次越高，曲线越尖锐；相反，松散化算子的幂次小于1，乘方运算后变大，隶属函数曲线趋于平坦化，而且幂次越高越平坦。

（2）模糊化算子

诸如“大约”、“近似”等这样的修饰词都属于模糊化算子，其作用是把肯定转化为模糊，如果对数字进行作用，就把精确数转化为模糊数。例如数字65是精确数，而“大约65”就是模糊数。如果对模糊值进行作用，就是模糊值的模糊，例如“年轻”是个模糊值，而“大约年轻”就变得更模糊。

在模糊控制中，采样的输入量总是精确量，要利用模糊逻辑推理方法，就必须首先把输入的精确量模糊化。模糊化实际上就是使用模糊化算子来实现的。所以，引入模糊化算子是非常有实用价值的。

（3）判定化算子

与模糊化算子有相反作用的另一类算子，例如“倾向于”、“偏向于”等，被称为判定化算子。其作用是可把模糊值进行肯定化处理，对模糊值作出倾向性判断。其处理方法有点类似于“四舍五入”，并常把隶属度为0.5作为分界线来判断。

由于语言变量适用于表达那些由于过分复杂而无法获得确定信息的概念和现象，它为这些通常无法进行定量化的“量”提供了一种近似处理方法。通过这种处理方法，就可把人的直接经验进行量化，转换成用计算机可以操作的数值运算。由此才能使人们有可能把专家的控制经验转换成控制算法并实现模糊控制。

7.4 模糊规则与模糊推理

7.4.1 模糊“如果—那么”规则

模糊规则是对自然或人工语言中的单词和句子定量建模的有效工具。通过将模糊规则理解为恰当的模糊关系，我们可以研究不同的模糊推理方案，用基于符合推理规则的推理

过程，从一组模糊规则和已知事实中推得结论。模糊规则和模糊推理是模糊推理系统的基础，是模糊集合理论最重要的建模工具，它们已被成功广泛地应用于各个领域。

模糊“如果—那么”规则（也称为模糊规则、模糊蕴含或模糊条件句）的形式为：

$$\text{如果 } x \text{ 是 } A\text{，那么 } y \text{ 是 } B \tag{7.19}$$

其中，A 和 B 分别是论域 X 和 Y 上的模糊集合定义的语言值。通常称“x 是 A”为前件或前提，“y 是 B”为后件或结论。

模糊系统或控制的基本单元是“如果—那么”规则：“如果水温偏高，那么就多加一些冷水”或“如果衣服很脏，那么洗涤时间应很长，否则洗涤时间不必太长”。一个模糊系统是一个“如果—那么”规则的集合，这些规则将输入映射到输出。每一个规则映射输入的一部分到输出的一部分。它将像“偏高的水温”这样的输入模糊集映射到像“较多的冷水”这样的输出模糊集。“偏高的水温”是模糊的，这是因为水的温度是“偏高”还是“偏低”都只是某种程度。一个单独的规则“如果水温偏高，那么就多加一些冷水”自身便定义了一个常数函数或线性段。叠合的规则可以定义出多项式或更复杂一些的函数。

模糊系统的发展是从词语到句子到段落。词意代表集合。名词“水温”代表着一个集合或水的一个子集。“水温”这个集合可能有模糊或灰色边界，但并非必需。形容词“高”和“很高”代表水温集合的模糊子集。因此，名词短语“偏高的水温”代表一个模糊集，“较多的冷水”这个名词短语同样也代表一个模糊集。我们可以用曲线来模拟这些模糊集，而不是用矩形，这些曲线可以是三角形、梯形或钟形曲线。像“如果水温偏高，那么就多加一些冷水”这样的一个完整句子，就代表着一个模糊规则或如果部分的模糊集“偏高的水温”和那么部分的模糊集“较多的冷水”之间的模糊联系。这样的一条规则定义了一个模糊输入/输出状态空间的一个子集，这种句子的列表定义了一个模糊系统或模糊规则集，这些集合覆盖了某个函数或某个函数族。增加或减少规则，就可对模糊系统进行宏观调控。

在我们使用模糊“如果—那么”规则对系统进行推理和分析之前，必须将表达式“如果 x 是 A，那么 y 是 B（或缩写为 $A \to B$）”的意义形式化。(7.19)式所描述的是两个变量 x, y 之间的关系，这意味着模糊“如果—那么”规则可以定义为乘积空间 $X \times Y$ 上的二元模糊关系 $\boldsymbol{R}$。

7.4.2 模糊逻辑推理

从已知条件求其结果的思维过程就是推理。用传统的二值逻辑进行假言推理和归纳推理时，只要大前提或者推理规则是正确的，小前提是肯定的，那么就一定会得到肯定的结论。即按照假言推理我们可以从 A 的真实性及其蕴含关系 $A \to B$ 推得 B 的真实性。例如，A 等于“西红柿是红的”，B 等于“西红柿是熟的”，如果“西红柿是红的”成立，那么“西红柿是熟的”也成立。以下过程说明了这个概念：

前提 1（规则）：　如果 x 是 A，那么 y 是 B

前提 2（事实）：　x 是 A

后件（结论）：　y 是 B

然而，在现实生活中，我们获得的信息往往是不精确、不完全的；或者事实本身就是模糊而不完全确定的，但又必须利用且只能利用这些信息进行判断和决策，此时，人们的推理是以近似的方式利用假言推理。

模糊逻辑推理是一种近似推理，它是从一组模糊“如果—那么”规则和已知事实中得出结论的推理过程。例如，假定有相同的蕴含规则“如果西红柿是红的，那么它是熟的”，而且已知“西红柿是或多或少有些红”，那么可以推得“西红柿是或多或少有些熟”。这可以表示为：

前提 1（规则）： 如果 x 是 A，那么 y 是 B

前提 2（事实）： x 是 A'

后件 （结论）： y 是 B'

其中，A' 接近于 A，B' 接近于 B，当 A，B，A' 和 B' 都是适当论域的模糊集合时，前面的推理过程被称为近似推理或模糊推理，也称为广义假言推理，因为假言推理是它的一个特例。同理，当存在事实“y 是 B'”时，通过前提 1 的规则，同样可以得出结论“x 是 A'”。

定义 7 近似推理（模糊推理）。

设 A, A' 和 B 分别是 X, X 和 Y 的模糊集合，模糊蕴含 $A \to B$ 表示为 $X \times Y$ 上的模糊关系 R，则由“x 是 A'”和模糊规则“如果 x 是 A，那么 y 是 B”导出的模糊集合 B' 定义为：

$$\begin{aligned}\mu_{B'}(y) &= \max_x \min[\mu_{A'}(x), \mu_R(x,y)] \\ &= \vee_x [\mu_{A'}(x) \wedge \mu_R(x,y)]\end{aligned} \tag{7.20}$$

或等价地：

$$B' = A' \circ R = A' \circ (A \to B) \tag{7.21}$$

由此，只要给模糊蕴含 $A \to B$ 定义好恰当的模糊关系，就可以采用模糊推理的步骤来求得结论了。

在模糊逻辑推理中，模糊前提 1 的规则“如果 x 是 A，那么 y 是 B”表示了 A 与 B 之间的模糊蕴含关系，记为 $A \to B$。对于模糊蕴含关系的运算方法，很多人进行了研究，常用的有以下两种方法。

①模糊蕴含最小运算（Mamdani）

$$R = A \to B = A \times B = \int_{X \times Y} \mu_A(x) \wedge \mu_B(y) / (x,y)$$

②模糊蕴含积运算（Larsen）

$$R = A \to B = A \times B = \int_{X \times Y} \mu_A(x) \mu_B(y) / (x,y)$$

模糊推理的结论是通过将事实与规则进行合成运算后得到。实际应用中广泛使用的合成运算方法也有以下两种。

1） 最大—最小合成法

$$\mu_{B'}(y)=\underset{x\in X}{\vee}\left[\mu_{A'}(x)\wedge\mu_R(x,y)\right]$$

2）最大—代数积合成法

$$\mu_{B'}(y)=\underset{x\in X}{\vee}\left[\mu_{A'}(x)\mu_R(x,y)\right]$$

其中，最大—代数积合成法常用在由模糊神经网络构造的模糊系统中。在模糊逻辑推理系统中，最常用的模糊推理方法是由英国伦敦大学的玛达尼（Mamdani）教授首先提出并使用的所谓最小—最大合成法。它是一种以模糊关系合成法则为基础的推理方法，它可以采用连续的模糊变量，也可以采用离散的模糊变量。模糊变量的隶属函数曲线可以是比较理想的钟形曲线，也可以是便于使用而相对比较简单的三角形。虽然在理论上钟形曲线比较理想，但是由于计算量大，在使用中十分不方便，因而一般用得很少。绝大多数都采用三角形。

设控制系统的控制规则格式为：

如果 $E=Ai$ 并且 $EC=Bj$ ，那么 $U=Cij$。

其中 $i=1,2,\ldots,m$, $j=1,2,\ldots,n$, 且 E 是偏差，Ai 是偏差的语言变量值；EC 是偏差变化率，Bj 是偏差变化率的语言变量值，Cij 是对应于 Ai 和 Bj 的控制量的语言变量值。则由模糊关系 R，可得：

$$R=\bigcup_{ij}Ai\times Bj\times Cij$$

其中 $i=1,2,\ldots,m, j=1,2,\ldots,n$，运算符 $\times$ 表示对模糊量求直积，即模糊关系的隶属函数为：

$$\mu_R(a,b,c)=\overset{i=m,j=n}{\underset{i=1,j=1}{\vee}}\mu_{Ai}(a)\wedge\mu_{Bj}(b)\wedge\mu_{Cij}(c)$$

其中 $\forall a\in A, \forall b\in B, \forall c\in C$, 且 A，B，C 分别是偏差、偏差变化率、控制量的论域。

对于特定的输入精确量 a^*, b^*, 则有输出：

$$U=(A\times B)\circ R$$

即

$$\mu_u(c)=\underset{a\in A,b\in B}{\vee}\mu_A(a^*)\wedge\mu_B(b^*)\wedge\mu_R(a,b,c)$$

最后再用质心法对 U 求精确值，即可得到最终的控制量。

这种处理方法的最大优点是控制规则中的前提和后件部分都与定性的语言描述对应，而且如上所示，可以把其推理过程用图形方式直观地表示出来，易于理解。但是用这种方法，往往所用的规则数目很多。一般而言，如果规则的条件部分的变量有 n 个 $x1....xn$，如果模糊标记取 7 个，可组合的规则数理论上为 7^n 个。其规则数是以幂次增加的。当然在实际控制系统中，并不是所有的规则数都会出现，有不少组合态是不会出现的，因而实际所需要的规则数只是这些组合数的一部分，甚至是一小部分。

下面将首先讨论上述定义中模糊推理的计算问题，然后进一步讨论在描述系统行为时具有多个前提的多条模糊规则的情况。鉴于最大—最小合成法应用广泛，且易于进行图

形解释，故以此为例进行讨论。

（1）具有单个前提的单一规则

这是最简单的情况，公式由（7.20）式给出。对此方程的进一步简化，有

$$\begin{aligned}\mu_{B'}(y) &= \vee_x[\mu_{A'}(x) \wedge \mu_R(x,y)] \\ &= \vee_x[\mu_{A'}(x) \wedge \mu_A(x) \wedge \mu_B(y)] \\ &= \vee_x[\mu_{A'}(x) \wedge \mu_A(x)] \wedge \mu_B(y) \\ &= \omega \wedge \mu_B(y)\end{aligned}$$

换言之，首先求出 $\mu_{A'}(x) \wedge \mu_A(x)$ 的最大值（图 7.6 中前件部分的阴影区域）；然后，结果 B' 的隶属函数就等于 B 的隶属函数被 ω 箝位后所得的图形，如图 7.6 中后件部分的阴影区域。直观上，ω 表示一条规则前件部分的可信度，这个测度由“如果—那么”规则传递，而且所求得的可信度或后件部分（图 7.6 中 B' ）的隶属函数不会大于 ω。

（2）具有多个前提的单一规则

具有两个前件的模糊“如果—那么”规则通常写作“如果 x 是 A 并且 y 是 B，则 z 是 C”，例如“如果压力偏高并且还在继续升高，那么停止加热”，相应的广义假言推理的问题可表示为：

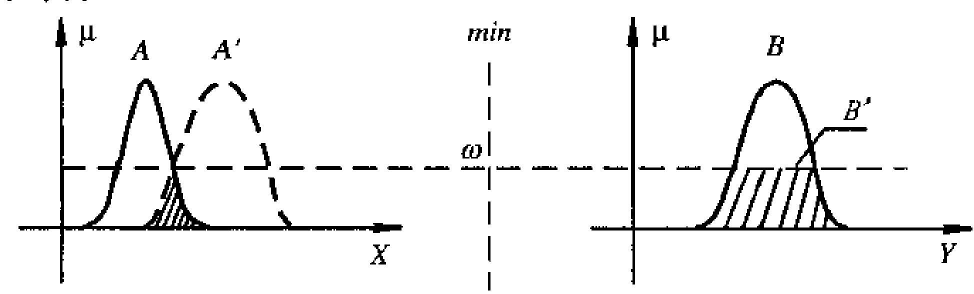


图 7.6 利用玛达尼蕴含及最大—最小法合成的广义近似推理的图形

解释论域为连续时具有单个前提及单一规则的推理过程

前提 1（规则）： 如果 x 是 A and y 是 B，那么 z 是 C

前提 2（事实）： x 是 A' and y 是 B'

后件 （结论）： z 是 C'

前提 1 中的模糊规则可以写成简单的形式“ $A\times B \to C$ ”。

这里 A 和 A' 、B 和 B' 、C 和 C' 分别是不同论域 X、Y、Z 上的模糊集合。

“如果 x 是 A and y 是 B，则 z 是 C”的数学表达式是：

$$\mu_A(x) \wedge \mu_B(y) \to \mu_C(z)$$

“x 是 A' and y 是 B' ”的意义是：

$$\mu_{A'\mathrm{and}B'}(x,y) = \mu_{A'}(x) \wedge \mu_{B'}(y)$$

若用玛达尼推理方法，用其蕴含定义 $A \to B = A \wedge B$ 作代换，就变成：

$$[\mu_A(x)\wedge\mu_B(y)]\wedge\mu_C(z)$$

由此可得推理结果 C' 为：

$$C'=(A\times B')\circ(A\times B\to C)$$

其隶属函数为：

$$\begin{aligned}\mu_{C'}^{(z)}&=\vee_{x,y}[\mu_{A'}(x)\wedge\mu_{B'}(y)]\wedge[\mu_A(x)\wedge\mu_B(y)\wedge\mu_C(z)]\\&=\vee_{x,y}\{[\mu_{A'}(x)\wedge\mu_{B'}(y)]\wedge\mu_A(x)\wedge\mu_B(y)]\}\wedge\mu_C(z)\\&=\{\vee_x[\mu_{A'}(x)\wedge\mu_A(x)]\}\wedge\{\vee_y[\mu_{B'}(y)\wedge\mu_B(y)]\}\wedge\mu_C(z)]\\&=(\omega_A\wedge\omega_B)\wedge\mu_C(z)\end{aligned}\tag{7.22}$$

其中，ω_A和ω_B分别是$A\cap A'$和$B\cap B'$隶属函数的最大值，通常ω_A表示 A 和 A' 之间的匹配度；ω_B类似。由于模糊规则的前件部分由连接词“与”组合而成，因此常称$\omega_A\wedge\omega_B$为模糊规则的激活强度，它表示规则的前提部分被激活的程度。推理过程见图7.7。与单输入情况一样，求出 A 对 A' 、B 对 B' 的隶属度ω_A、ω_B，并且取这两个之中小的一个值作为总的模糊推理前提的隶属度，再以此为基准去切割推理结论的隶属函数，结果 C' 的隶属函数等于 C 的隶属函数被激活强度$\omega_C(\omega_C=\omega_A\wedge\omega_B)$箝位后的结果。这个结论可以直接推广到多于两个前提的情况。

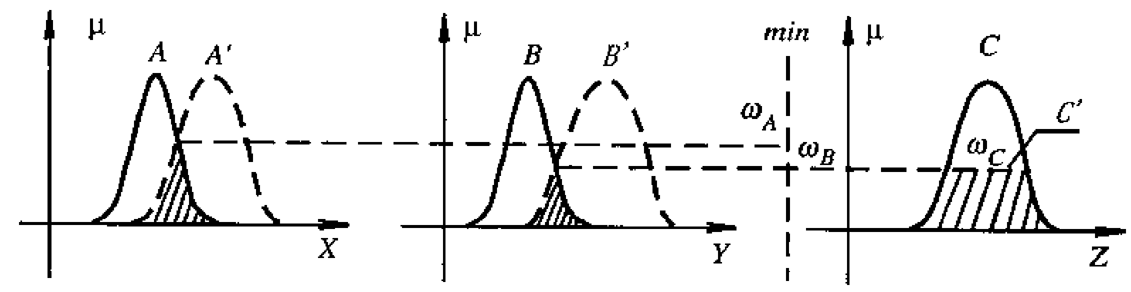


图 7.7 多个前提中单一规则推理过程的图形解释

(3) 具有多个前提的多条规则

两输入的情况很容易就可推广到多输入的情况。对于具有多个前提的多条规则，通常处理为相应于每条模糊规则的模糊关系的并集。因此，其广义假言推理问题可以写为：

前提 1（规则）： 如果 x 是 A_1 and y 是 B_1，那么 z 是 C_1

前提 2（规则）： 如果 x 是 A_2 and y 是 B_2，那么 z 是 C_2

前提 3（事实）： x 是 A' and y 是 B'

后件 （结论）： z 是 C'

我们可以按照图 7.8 所示的推理过程来求得输出结果模糊集合 C' 。只要分别先求出

各个输入对推理前提中相应条件的隶属度，再取其中最小的一个作为总的模糊推理前件的隶属度，去切割推理结论的隶属函数，便可得到推理的结论。

为了验证这一推理过程，令 $R_1 = A_1 \times B_1 \to C_1$ 、$R_2 = A_2 \times B_2 \to C_2$ 由于最大—最小算子“$\circ$”是对并算子“$\cup$”进行分配，则有

$$\begin{aligned} C' &= (A' \times B') \circ (R_1 \cup R_2) \\ &= [(A' \times B' \circ R_1)] \cup [(A' \times B' \circ R_2)] \\ &= C'_1 \cup C'_2 \end{aligned}$$

如果是多输入又是多推理规则的情况，又如何进行推理呢？以两输入多规则情况为例，若有 n 条规则，其一般形式为：

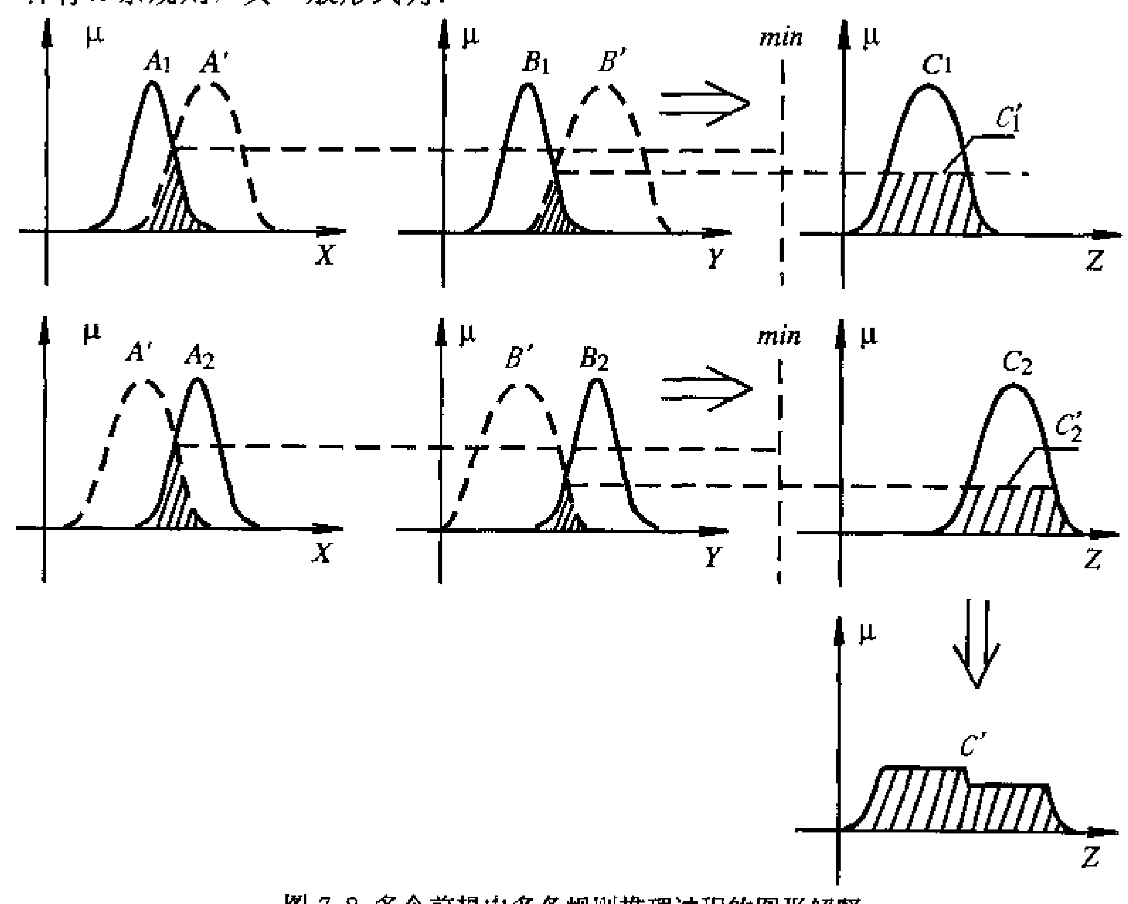


图 7.8 多个前提中多条规则推理过程的图形解释

前提 1（规则）： 如果 x 是 A_1 and y 是 B_1，则 z 是 C_1

前提 2（规则）： 如果 x 是 A_2 and y 是 B_2，则 z 是 C_2

……

前提 n（规则）： 如果 x 是 A_n and y 是 B_n，则 z 是 C_n

前提 $n+1$（事实）： x 是 A' and y 是 B'

后件 （结论）： z 是 C'

这里，A_i和 A' 、B_i和 B' 、C_i和 C' 分别是不同论域 X、Y、Z 上的模糊集合。

“A_i 且 B_i”的意义是：

$$\mu_{A_i\text{and}B_i}(x,y)=\mu_{Ai}(x)\wedge\mu_{Bi}(y)$$

“如果 A_i 且 B_i ，那么 C_i ”的数学表达是：

$$\mu_{A_i}(x)\wedge\mu_{B_i}(y)\to\mu_{C_i}(z)$$

若用玛达尼推理方法，用其蕴含定义 $A\to B=A\wedge B$ 作代换，就变成：

$$[\mu_{A_i}(x)\wedge\mu_{B_i}(y)]\wedge\mu_{C_i}(z)$$

由此得推理结果为：

$$\begin{aligned}C&=(A\ ANDB)\circ\{[(A_1ANDB_1)\to C_1]\cup\\&\quad\cdots\cup[(A_nANDB_n)\to C_n]\}\\&=\{(A\ ANDB)\circ\{[(A_1ANDB_1)\to C_1]\}\cup\\&\quad\cdots\cup\{(A\ ANDB)\circ[(A_nANDB_n)\to C_n]\}\\&=C_1\cup C_2\cup\cdots\cup C_n\end{aligned}$$

其中：

$$\begin{aligned}C_i&=(A\ ANDB)\circ\{[(A_iANDB_i)\to C_i]\\&=[A\circ(A_i\to C_i)]\cap[B\circ(B_i\to C_i)],\end{aligned}\qquad(I=1,2,\ldots,n)$$

其隶属函数为：

$$\begin{aligned}\mu_{Ci}(z)&=\underset{X}{\vee}\{\mu_A(x)\wedge[\mu_{A_i}(x)\wedge\mu_{C_i}(z)]\}\cap\underset{X}{\vee}\{\mu_B(y)\wedge[\mu_{B_i}(y)\wedge\mu_{C_i}(z)]\}\\&=\underset{X}{\vee}\{\mu_A(x)\wedge\mu_{A_i}(x)\}\wedge\mu_{C_i}(z)\cap\underset{X}{\vee}\{\mu_B(y)\wedge\mu_{B_i}(y)\}\wedge\mu_{C_i}(z)\\&=(\omega_{A_i}\wedge\mu_{C_i}(z))\cap(\omega_{B_i}\wedge\mu_{C_i}(z))\\&=(\omega_{A_i}\wedge\omega_{B_i})\wedge\mu_{C_i}(z)\end{aligned}$$

如果有两条两输入的规则，那么得到两个结论：

$$\mu_{C_1}(z)=\omega_{A_1}\wedge\omega_{B_1}\wedge\omega_{C_1}$$
$$\mu_{C_2}(z)=\omega_{A_2}\wedge\omega_{B_2}\wedge\omega_{C_2}$$

其意义为分别从不同的规则得到不同的结论，几何意义是分别在不同规则中用各自推理前提的总隶属度去切割本推理规则中结论的隶属函数，以得到输出结果。再对这所有的结论求模糊逻辑和，即进行“并”运算，便得到总的推理结论。

$$C=C_1\cup C_2\cup\cdots\cup C_n$$

这种推理方法是先在推理前提中选取各个条件中隶属度最小值（即“最不适配”的隶属度）作为这条规则的适配程度，以得到这条规则的结论，这称为取小（min）操作；再对各规则的结论综合，选取其中适配度最大的部分，即取大（max）操作。整个并集的面积部分就是总的推理结论。这种推理方法简单，得到广泛应用。但是它也有一个缺点，那就是其推理结果经常不够平滑。由此，有人主张把从推理前提到结论削顶法的“与”运算改成“乘积”运算，这就不是用推理前提的隶属度为基准去切割推理结论的隶属函数，而

是用该隶属度去乘结论的隶属函数，这样得到的结论就不是呈平台梯形，而是原隶属函数的等底缩小。这种处理结果经过对各个规则结论“并”运算后，总推理结果的平滑性得到改善。

在实际应用中，常常采用输入量的模糊集合为模糊单值（singleton），即 $A'=\frac{1}{x_0}$，$B'=\frac{1}{y_0}$，则有：

$$\mu_C(z)=\bigcup_{i=1}^{l}\omega_i \wedge \mu_{C_i}(z)$$

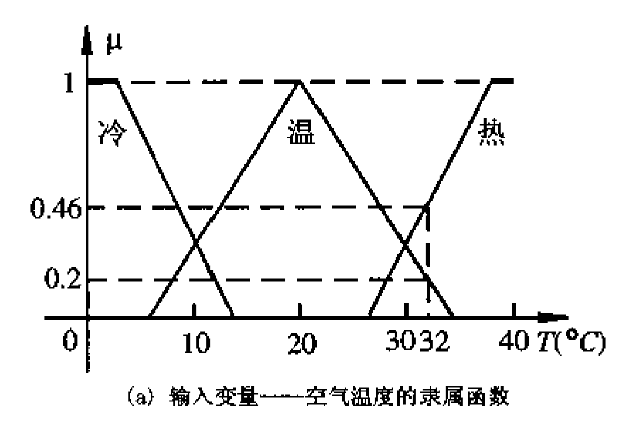
其中

$$\omega_i=\mu_{A_i}(x_0)\wedge \mu_{B_i}(y_0)$$

这里，ω_i 可以被看成是相应于第 i 条规则的加权因子，也可以被看成是第 i 条规则的适用程度或其对模糊作用所产生的贡献大小。下面给出了一个采用离散的单值进行模糊推理的简单实例。

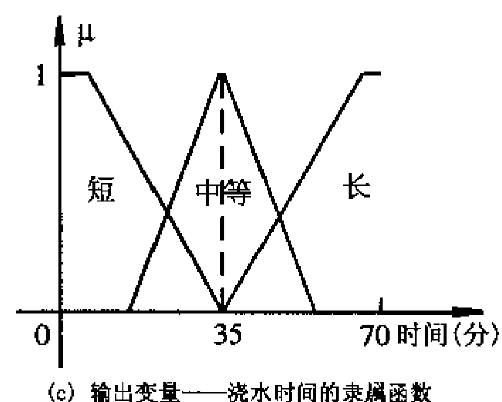
例 7.3 采用模糊逻辑推理的方法，根据室外温度以及草地干湿状况，对室外草地进行自动喷水控制量的具体求解。

首先确定输入/输出论域及其隶属函数。这是一个自动喷灌系统的设计问题。系统的输入变量为温度和湿度；控制量是浇水量的大小。



(a) 输入变量——空气温度的隶属函数

(b) 输入变量——土壤湿度的隶属函数



(c) 输出变量——浇水时间的隶属函数

图 7.9

我们定义空气温度的论域为 0℃到 40℃。这里之所以把下限定为 0℃，是因为喷灌系统在水冻结冰时是不能工作的。据此，我们主观地把空气温度模糊变量的论域定义为：

空气温度：[0，40]，模糊标记为：冷、温、热。

与此类似，对土壤湿度的论域定义为：

土壤湿度：[0%，30%]，模糊标记为：干、适中、湿。

它们的隶属函数如图 7.9 所示。

1）设控制规则为

湿度＼温度	冷	温	热
干	中等	长	长
适中	短	中等	长
湿	短	中等	中等

对于任意输入变量，如：温度 ＝32℃，湿度 ＝11%。

这里，对输入变量采用单值模糊化，由此在各自隶属函数图（见图 7.9(a)和(b)）上共可得如下四条控制规则：

①“如果温度是热（0.46）且土壤是干（0.25），那么加水时间为长”；

②“如果温度是温（0.2）且土壤是适中（0.75），那么加水时间为中等”；

③“如果温度是温（0.2）且土壤是干（0.25），那么加水时间为长”；

④“如果温度是热（0.46）且土壤是适中（0.75），那么加水时间为中等”。

实际上，因为各模糊标记之间的隶属函数相交不超过两个，所以无论模糊标记取多少，对于任意输入温度和湿度，都只与两个模糊标记相关，所以都只有四条控制规则被激活。

2）控制规则的强度

一旦确定了每一前提的相关隶属度的值，则要确定每一控制规则的强度。既然前提是由“且”即“与”也就是“and”来联接的，所以规则强度的选择是以前提中的最小强度值为该规则的真值。对于前面四条规则，各自的规则强度分别为：

① min(0.46,0.25) = 0.25；

② min(0.2,0.75) = 0.2；

③ min(0.2,0.25) = 0.2；

④ min(0.46,0.75) = 0.46。

3）根据规则强度来确定模糊输出

由模糊规则控制表以及②中所确定的规则强度值可获得控制量的隶属度，即：

“如果温度是热（0.46）且土壤是干（0.25），那么加水时间为长（0.25）”；

“如果温度是温（0.2）且土壤是适中（0.75），那么加水时间为中等（0.2）”；

“如果温度是温（0.2）且土壤是干（0.25），那么加水时间为长（0.2）”；

“如果温度是热（0.46）且土壤是适中（0.75）,那么加水时间为中等（0.46）”。

由此得出输出控制量在确定输入温度和湿度下的模糊控制变量的隶属度。并且从中可以看到，在不同输入前提条件下，得到相同的控制行为（如控制规则①和③具有相同的控

制规则为“加水时间为长”的控制，控制规则②和④具有相同的控制规则为“加水时间为中等”的控制)。在这种情况下，模糊控制的输出为具有相同输出行为的所有规则中规则强度最大的那个。在本例中，可得输出控制模糊量为加水时间为长（0.25）和中等（0.46）。换句话说，有两个或两个以上的相同控制企图影响输出时，规则强度为最大的那个值起主导作用。最后控制输出量为：

$$u=\frac{0.25*70+0.46*35}{0.25+0.46}=\frac{17.5+16.1}{0.71}=47.32$$

通过以上过程，可以得出一个模糊推理系统的求解步骤为：

①建立控制规则；

②确定前提的真值：对于每一精确输入值，通过使用隶属函数进行模糊化处理，以获得每一前提的真值（隶属度）；

③求出每一控制规则的强度：它等于每一规则中最小的前提隶属度；

④确定每一结论标记的模糊输出：它等于相同结论标记中的最大规则强度；

⑤求模糊输出控制量。

以上计算出的只是在一组输入下的输出，从中可以看出其计算的复杂性。这也正是简单的模糊逻辑控制往往是事先做好控制律的表格存储到计算机中的原因，当然，由于内存有限，必然导致控制精度不高的情况出现。要想提高模糊逻辑控制的效率和控制精度，有效的办法是引入神经网络的自适应功能。这将在后面的章节中进行详细讨论。

第 8 章 模糊控制器的设计方法

无论是采用自动控制原理，还是现代控制理论来设计一个自动控制系统，都需要事先建立被控对象的数学模型，需要知道模型的结构、阶次和参数等方面的情况，在此基础上合理地选择控制策略，进行控制器的设计。但是在实际系统的控制中，由于被控对象的复杂性，很难建立起精确的数学模型，这时，采用经典控制理论就无法得到满意的控制效果，甚至无法对系统进行控制。

在很多类似于以上的现实控制系统中，采用模糊逻辑控制常常能够达到良好的控制效果。模糊控制的基本思想就是利用计算机来实现人的控制经验，而人的控制经验一般是由语言来表达的，语言控制规则通常用“如果 A，那么 B”的方式来表达在实际控制中的专家经验和知识，所以模糊控制规则往往具有相当的模糊性，其最大特征是将专家的控制经验、知识表示成语言控制规则，然后用这些规则去控制系统。因此，模糊控制特别适用于模拟专家对数学模型未知的、复杂的、非线性系统的控制。这些控制规则是模糊条件语言的形式，可以用模糊数学的方式来描述过程变量和控制作用等模糊概念及它们之间的关系，然后又可以由这些关系某时刻的过程变量值采用模糊推理的方法求出此时刻的控制量。

8.1 精确与模糊控制的事例

为了更好地说明模糊逻辑控制，首先用线性和模糊这两种不同的方法来解决同一个问题。在此所选择的是一个比较能够说明问题的小费问题。在国外餐馆吃饭，一般都要给侍者付服务小费，基本的付小费问题可以这样叙述：用 0 到 10 表示餐馆的服务质量（其中 10 表示最好），那么小费应当如何付？此处需说明的是，在国外，尤其在美国，在餐馆中就餐的平均小费应当占餐费的 15%。当然，实际付小费的多少是根据服务质量而有所变化的。下面我们分别以不同的方式来看看付小费的情况。

8.1.1 采用精确的非模糊求解方法

最简单的付小费的关系式可以是按账单费用的 15%来付给。如果我们将账单数定义为 1，那么小费金额则为：

$$小费 = 0.15$$

服务质量与付费之间的关系式可以用图形表示为如图 8.1 所示。

很显然，这种关系式并没有真正考虑服务的质量，给人感觉有些不合理，应当根据服务质量的好坏来变化给小费的数量。在同时考虑平均小费为 15%的情况下，我们不妨只给最次的服务 5%的小费，而随着服务质量的增加而线性地从 0 增加到 10，以使最差的服

务小费为 5%，最好的小费为 25%，其数学关系式可以写为：

小费 ＝0.20/10 * 服务质量 ＋0.05

此时的小费与服务的关系图如图 8.2 所示。

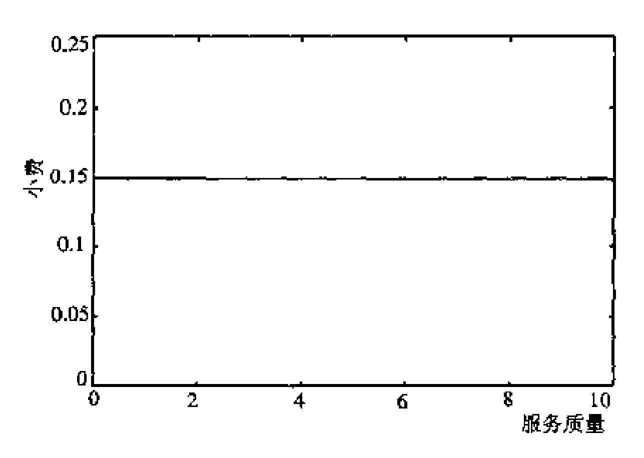


图 8.1 服务质量与付费之间的关系图

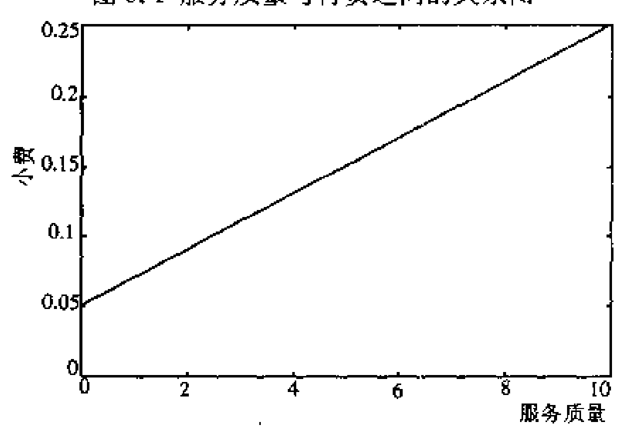


图 8.2 小费与服务的关系图

现在看来似乎合我们的意。其小费的付法和服务质量挂上了钩，而且很直观。更深层次地我们会想到，我们的满意度不仅取决于服务质量，同时与餐厅所供食物的质量也是密不可分的。因为如果食物质量很差，服务质量再好我们也不会满意。所以，更合理的是应当同时考虑服务与食品两个方面的质量。由此将付小费的关系式从仅取决于服务质量的一维变量，扩展到同时考虑服务以及食品质量的二维关系式。同样我们定义食品质量的等级也是从 0 到 10（其中 10 表示极好）。沿用上面的图 8.2 表示的变化的关系式，我们可以写出新的付小费的关系式为：

小费 ＝0.20/20 * (服务质量+食物质量)＋0.05

其关系图如图 8.3 所示。

很好，这是一种做法。不过仔细想想，虽然付小费应当同时考虑服务与食品这两个因素，不过似乎服务质量的因素更为重要，小费主要付的是服务质量费。这样想来，在这两个因素中，应将服务质量在小费中占较大的比重。如让服务质量在整个小费中占 80%，而食品质量只在小费中占 20%，所以可以重新写出如下算式：

服务比率 ＝0.8；

小费 = 服务比率*（0.20/10*服务质量+0.05）+（1–服务比率）*（0.20/10*食品质量+0.05）

其关系图如图 8.4 所示。

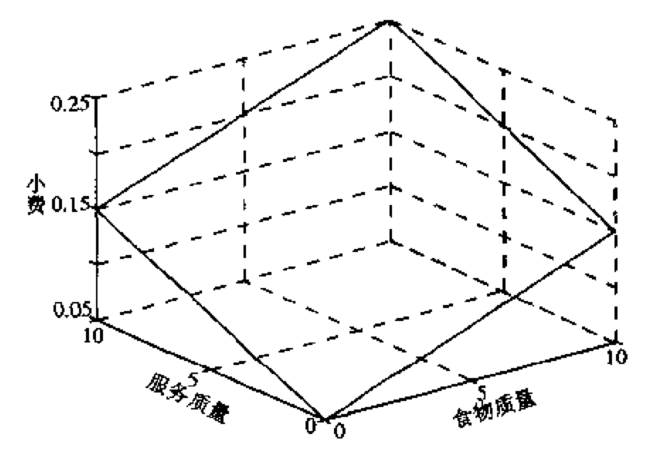


图 8.3 小费与服务质量及其食物质量的关系图

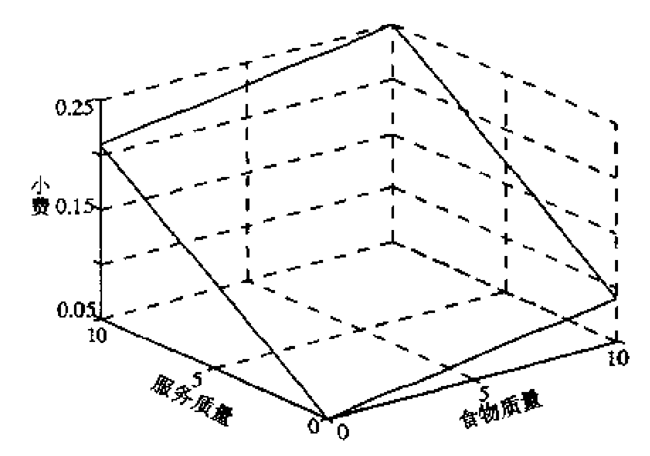


图 8.4 小费与服务质量及其食物质量的关系图

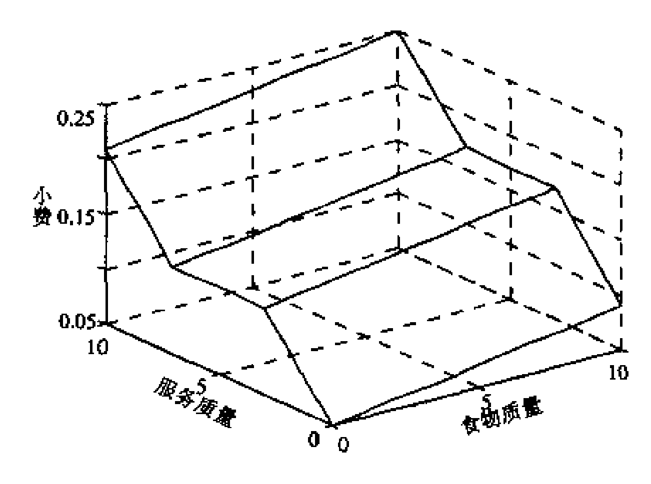


图 8.5 小费与服务质量及其食物质量的关系图

从整体上看，其小费的付出似乎又太线性增长了，总感觉在中间一段应当平缓一些，即希望在一般的情况还是付常规的 15%，只是在服务太糟或服务极好时在付费上有所差别，以表示自己的满意度。我们仍然可以用线性关系来达到此目的，但要用几个条件语句：

如果 0≤ 服务质量< 3，小费 =（0.10/3）* 服务质量+0.05；

如果 3 ≤ 服务质量< 7，小费 =0.15；

如果 7 ≤ 服务质量 ≤ 10，小费 =（0.10/3）*（服务质量-7）+0.15。

而对食物质量的关系式不做任何变动。这需要再在上面的条件句中加上以前的食物关系式，由此可得到付小费的最终关系图如图 8.5 所示。

现在的图形看起来十分令人满意。不过回过头来想一想，从最初的想法，经过五次改进，最后的小费计算公式已变得相当复杂。如果想进一步改进，算式还将变得更复杂。而且对于不是从最新设计思想过来的人很难看清其算法是如何工作的。下面我们再来看看用模糊方法是如何解决问题的。

8.1.2 模糊方法

如果只需要抓住问题的实质就可以解决问题，那是最好不过的了。对于前面的小费问题，我们列出问题的实质性内容，可以归纳为：

①如果服务质量不好，那么小费低；

②如果服务质量好，那么小费中；

③如果服务质量极好，那么小费高。

上述规则的表达顺序可以是任意的，哪一条在前在后是无关紧要的。如果我们想把食物的质量因素也包括在内，可以另加下列两条规则：

④如果食物质量差，那么小费低；

⑤如果食物可口，那么小费高。

实际上，我们可以将两种不同的规则合并成三条规则：

①如果服务质量不好或者食物质量差，那么小费低；

②如果服务质量好，那么小费为中；

③如果服务质量极好或者食物可口，那么小费高。

这三条规则就是我们解题的核心。这些规则正好构成了模糊逻辑系统的规则。同时采用模糊逻辑推理还具有其他特点，如很容易增加规则而不影响已有的规则以及模糊规则具有普遍的适应性等。现在只要我们给语言变量赋予数学定义（比如小费“中”是多少），就可以得到一个完整的模糊推理系统。当然，这里面还涉及到模糊推理方法。比如，现有的规则如何合并，如何定义语言变量等，这些都是模糊逻辑推理的细节。不过这些具体的方法是具有通用性的，它们不会因问题的改变而有很大的变化。模糊逻辑推理的通用性使模糊逻辑应用起来十分地方便，同时它还具有自适应性、简单和易于应用等特点。

这里我们给出在上述三条规则下的模糊逻辑系统给出的有关小费问题的答案，如图 8.6 所示。从中可以看出，如果采用线性分段来表示将是很复杂的，但用模糊逻辑推理三条规则足矣，而且相当准确地表达了人的意愿。

为了更好地认识和掌握模糊逻辑推理系统，下面我们从模糊逻辑控制过程入手。

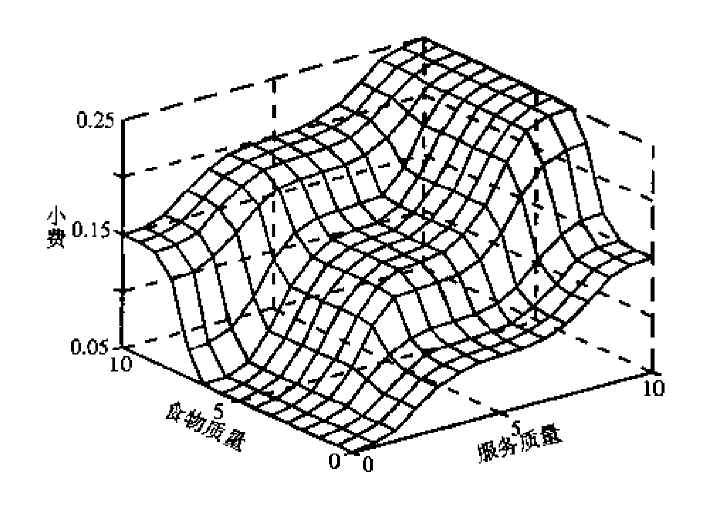


图 8.6 小费问题的输入输出关系图

8.2 模糊逻辑控制过程

一般而言，模糊控制是建立在人的经验和常识的基础上，这就是说，操作人员对被控系统的了解不是通过精确的数学表达式，而是通过操作人员丰富的实践经验和常识。由于人的决策过程本质上就具有模糊性，因此，控制动作并非稳定一致，且有一定的主观性。但是，有经验的模糊控制设计工程师可以通过对操作人员控制动作的观察和与操作人员的交谈讨论，用语言把操作人员的控制策略描述出来，以构成一组用语言表达的定性的决策规则。如果把那些熟练技术工人或者技术人员的实践经验进行总结和形式化描述，用语言表达成一组定性的条件语句和精确的决策规则，然后利用模糊集合作为工具使其定量化，设计一个控制器，用形式化的人的经验法则模仿人的控制策略，再驱动设备对复杂的过程进行控制，形成模糊控制器。

由一种模糊规则构成的模糊系统可代表一个输入、输出的映射关系。从理论上说，模糊系统可以逼近任意的连续函数。要表示从输入到输出的函数关系，模糊系统除了模糊规则外，还必须有模糊逻辑推理和解模糊的部分。模糊逻辑推理就是根据模糊关系合成的方法，从数条同时起作用的模糊规则中，按并行处理方式产生对应输入量的输出模糊子集，解模糊过程则是将输出模糊子集转化为非模糊的数字量。图 8.7 给出了一般模糊推理系统的方框图。

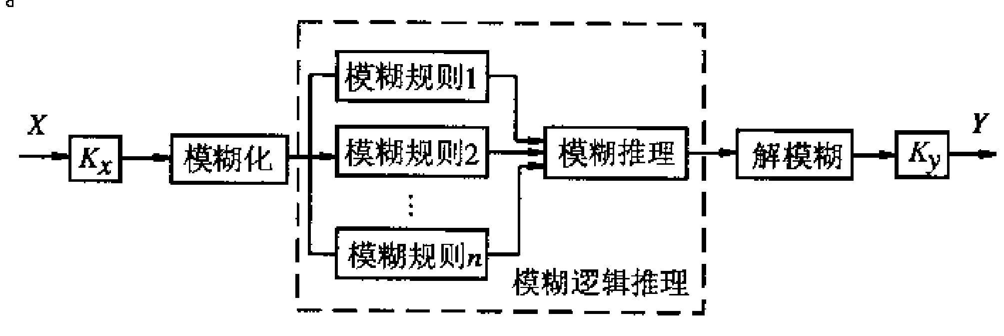


图 8.7 模糊推理控制器的方框图

模糊控制系统的核心是模糊逻辑控制器，模糊逻辑控制器一般是可以靠软件编程来实

现的。实现模糊逻辑控制的一般步骤如下：

第一步，通过传感器把要监测的物理量变成电量，再通过模数转换器把它转换成精确的数字量，精确数字量输入经过模糊逻辑控制器，首先把这个精确的输入量转换成模糊集合的隶属函数，这一步就是精确量的模糊化；

然后，根据有经验的操作者或者专家的经验制定出模糊控制规则，并进行模糊逻辑推理，以便得到一个模糊输出集合，即一个新的模糊隶属函数，这一步为模糊控制规则形成和推理，其目的是利用模糊输入值获取适当的控制规则，为每个控制规则确定适当的隶属度，并且通过加权计算合并那些规则的输出。

最后，根据模糊逻辑推理得到的输出模糊隶属函数，用不同的方法找一个具有代表性的精确值作为控制量，这一步成为模糊输出量的解模糊判决，其目的是把分布范围概括合并成单点的输出值，加到执行器上实现控制。一般模糊控制系统的方框图如图 8.8 所示。

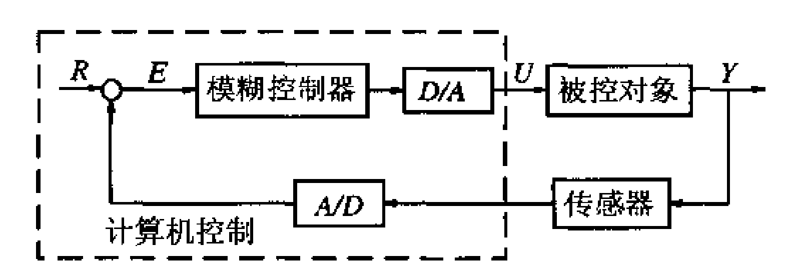


图 8.8 模糊控制系统的方框图

在模糊控制系统设计中，怎样设计和调整模糊控制器及其参数是一项很重要的工作。根据以上对模糊逻辑控制过程以及模糊控制系统的描述可知，设计模糊控制器主要包括以下几项内容：

①确定模糊控制器的输入变量、输出变量和论域；

②确定模糊化和解模糊化方法；

③确定模糊控制器的控制规则及其模糊推理方法；

④量化因子及比例因子的选择；

⑤编制模糊算法的应用程序；

⑥系统的仿真实验及参数的确定。

8.3 输入变量和输出变量的确定

模糊控制器的结构设计就是要确定模糊控制器的输入变量和输出变量。究竟选择何种信息作为模糊控制系统变量，必须深入研究手动控制过程中有经验的操作人员主要根据哪些信息控制被控对象向预期目标逼近。

人在进行手动控制过程中，操作者期望实现控制目标，一旦偏离目标，出现了偏差，操作者便根据偏差大小进行调整。人的大脑中误差的“大”和“小”，这些概念是模糊的。在整个手动控制过程中，人所能获得的信息一般可以概括为三个：误差、误差的变化和误差变化的速率。在手动控制过程中，人对误差、误差的变化以及误差变化的速率这三种信息的敏感程度是完全不同的。由于模糊控制器的控制规则往往是根据手动控制的大量实践

经验总结出来的，因此，模糊控制器的输入变量自然也可以有三个：误差、误差的变化和误差变化的速率；而输出变量一般选择为控制量或控制量的变化，即增量。

通常将模糊控制器输入变量个数称为模糊控制的维数。常见的模糊控制器的结构有三种形式，如图 8.9 所示。从理论上讲，模糊控制器的维数越高，控制效果也越好，但是维数高的模糊控制器实现起来相对于维数低的要复杂和困难得多。人们经常使用的是二维模糊控制器。在本书不加说明的控制器设计中，都是以误差和误差的变化作为控制器输入变量的。

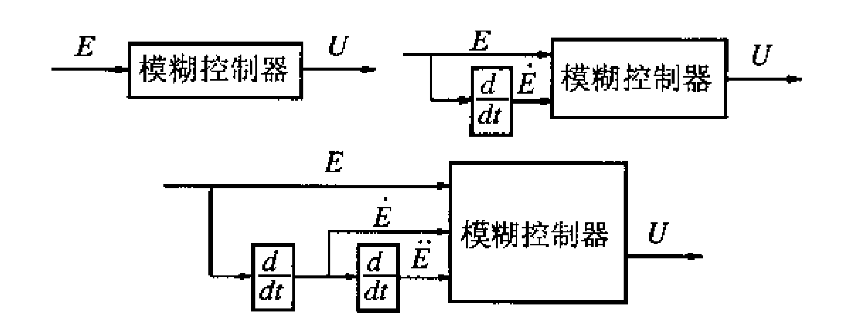


图 8.9 常用的模糊输入变量，其中 E 为误差

8.4 论域的确定

模糊控制器的输入信号（误差、误差变化率）的实际范围称为这些变量的论域，为了确定论域，首先应该确定整个设计系统相联系的变量所应用的范围。这个选择范围应该是经过细心地推敲过的。例如如果指定的范围太小，那么正常的数据就可能会出现在所定义的论域之外，由此所得系统的性能就可能受到影响。反之，如果定义的基本论域太大，就会对某些数值响应迟钝。这对某些具有饱和现象的系统没有问题，但是在其他系统中就会出现问题。同样，每个输出变量的论域范围也应该仔细地推敲过。一般而言，如果论域被定义得太大也会出现问题。在这种情况下，控制器工作过程中的不被使用区域就比较大。这个问题在解模糊时就会更加明显，因为这极不均匀面积很大的隶属函数就会使它在质心法中产生较大的偏移，使其他与其交叉的隶属函数的影响不适当地减小。

论域离散化经常被称为量化。实际上，量化是将一个论域离散成确定数目的几小段（量化集），每一段用某一个特定术语作为模糊标记，这样就形成一个离散比例。然后通过对这新的离散域中每个特定术语赋予隶属度来定义模糊集。为减少控制器运行时间，可用离线处理方式先完成建立在离散化上的控制查询表，用该表定义对所有可能的输入信号进行合并的控制器输出。在模糊逻辑控制是连续的情况下，量化等级的数目应该足够大，以便有充分的近似度，但另一方面，又要尽量减少该等级数以减少复杂性和存储量。量化等级的选择对控制效果具有关键性的作用。为实现离散化，就需要做标记映射，把测量变量转换成离散论域中的量值，这种映射可用区间均匀（线性）关系，也可用非均匀（非线性）关系，或者两者兼而有之。量化等级的选择与某些先验知识有关。例如用粗糙的解决方法其误差就大，而用清晰的解决方法其误差就小。一般而言，由于离散化，将影响模糊控制

器的性能，并对过程状态变量值的小偏差将更不灵敏。

为实现模糊控制器的标准化设计，目前在实际中常用的处理方法是把误差的变化范围设定为[-6, +6]区间连续变化量(或根据所选取的模糊标记数来确定论域的范围)，使之离散化，以论域范围取[-6,+6]为例，构成含有 13 个整数元素的离散集合：

$$\{-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6\}$$

实际上如果是非对称型的，也可用 1～13 取代-6～+6。

如上所述，实际系统工作的精确输入量的变化范围一般不会是在[-6,+6]之间，如果其基本论域是在[a, b]之间的话，可以通过变换：

$$y = \frac{12}{b-a}\left[x - \frac{a+b}{2}\right] \tag{8.1}$$

将在[a, b]之间变化的变量 x 转化为[-6,+6]之间变化的变量 y。

模糊逻辑控制中有关变量名的定义如图 8.10 所示。

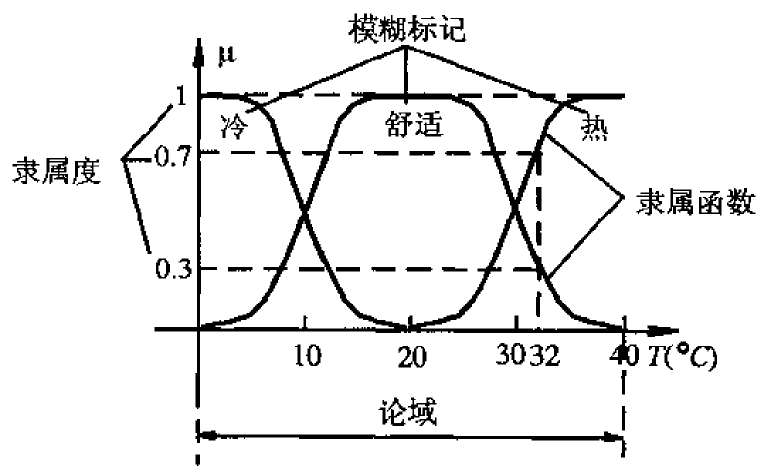


图 8.10 模糊逻辑控制中变量名的定义图解

论域中的量是精确量。通常可以定义误差的基本论域为[$-m$, m]，误差变化的基本论域为[$-n$, n]。模糊控制器的输出信号（控制量）的实际变化范围是控制量的基本论域，通常定义为[$-l$, l]，它也是精确量。M, n, l 分别为正整数。为了确保各模糊子集能具有较好的覆盖率，避免出现失控现象，通常要求 $m \geq 6$, $n \geq 6$, $l \geq 7$。从理论上讲，增加论域中的元素个数，可以提高控制精度，但也带来了计算量增大，占用内存增多等不利因素。实际经验表明，按等级分得过细并没有太大必要。对论域选择由于在系统调试时对被控对象缺乏足够的经验知识，因此只能作初步的确定，在实际调试时再加以具体认定。

8.5 确定模糊化和解模糊化方法

由模糊控制器的结构可知，控制器的输入和输出信号均是精确量，而进行模糊推理需要模糊量，这样就需要在模糊控制算法实现过程中，能够进行精确量与模糊量之间的相互转换。

8.5.1 模糊化方法

将精确量（实际上是数字量）转化为模糊量的过程称为模糊化或称为模糊量化。模糊化与自然语言的含糊和不精确相联系，这是一种主观评价，它把测量值转换为主观量值的评价。由此，它可定义为在确定的论域中将观察到的输入空间转换为模糊集的映射。在处理本质上无论是主观还是客观的不确定信息中，模糊化都扮演着重要角色。

模糊控制规则中前提的语言变量构成模糊输入空间，结论的语言变量构成模糊输出空间。每个语言变量的取值为一组模糊语言名称，称为模糊标记。它们构成了语言名称的集合。每个模糊标记对应一个模糊集合，对于每个语言变量，所取值的模糊集合都具有相同的论域。这些模糊标记通常均具有一定的含义。一般而言，“大、中、小”三个词汇常被人们用来描述输入和输出变量的状态。由于人的行为在正、负两个方向的判断基本是对称，将大、中、小再加上正、负两个方向（极性）并考虑零状态，这样一共就有 7 个词汇，可组成 7 个模糊标记，即：

{负大，负中，负小，零，正小，正中，正大}

或用英文字头书写的形式表示为：

{NB，NM，NS，Z，PS，PM，PB}

对误差的变化这个输入变量，在选择描述其状态词汇时，常常将“零”分别为“正零”和“负零”，以表示误差的变化在当前是“增加”趋势还是“减少”趋势。于是词集又增加“正零”（PZ）和“负零”（NZ）。

一般采用 7 个～11 个模糊标记把连续变量分成有限的若干档，每一档表示该变量的一种模糊状态。有时也可设有 5 个模糊状态，甚至是采用 3 个模糊状态。所用模糊状态越少，模糊控制规则的条件组合数目就越少，从总体上说，控制算法就越简单而粗略。反之设定的模糊状态数越多，其规则数就越多，规则制定就更细致，在模糊规则设计合适的情况下，控制就可更平滑；但是同时规则变得更复杂，并且运算量加大，由此在处理器的运算速度不足够快的情况下，就有可能影响控制的实时性。所以在确定模糊标记时，要根据目标兼顾考虑或通过计算机模拟仿真来确定。

描述输入和输出变量词汇都具有模糊特性，可用模糊集合来表示。因此，模糊集合的确定问题就转化为求取模糊集合隶属函数的问题了。

由于模糊变量没有明确的外延，如何用具体数据来刻画一个模糊变量的性质，这就是模糊子集的确定问题。对模糊子集的理想要求是它必须客观地反映实际情况。

定义一个模糊子集，实际上就是要确定模糊子集隶属函数曲线的形状。将确定的隶属函数曲线离散化，就得到有限个点上的隶属度，便构成了一个相应的模糊变量的子集。统计结果表明，用正态型模糊变量来描述人进行控制活动的模糊概念是比较适宜的。并且，隶属函数曲线形状较尖的隶属模糊子集，其分辨率较高，控制灵敏度也高；相反，隶属函数曲线形状较平缓，控制特性也就比较平缓，稳定性能也较好。因此，在选择模糊变量的隶属函数时，在误差较大的区域采用低分辨率的模糊集，在误差较小的区选用较高分辨率的模糊集，在误差接近于零时选用高分辨率的模糊集，这样才能达到控制精度高而稳定性好的控制效果。

从自动控制的角度来看，希望一个控制系统在要求的范围内都能够很好实现控制。模糊控制系统设计时也要重视这个问题。因此，在选择描述某一模糊变量的各个模糊子集时，要使它们在整个论域上的分布合理，即它们应该较好地覆盖整个论域。隶属函数的分布原则应当是：

①论域中的每个点应该属于至少一个隶属函数的区域，同时它一般应该属于不多于两个隶属函数的区域；

②对同一个输入没有两个隶属函数同时具有最大隶属度；

③当两个隶属函数重叠时，重叠部分的任何点的隶属函数之和应该不大于 1。

在模糊控制应用中，被观察量通常是确定的量。因为在模糊逻辑控制中的操作是基于模糊集合理论，因此首先必须先进行模糊化。模糊逻辑控制的设计经验为我们提出模糊化的主要方法有以下两种。

1）精确量离散化

精确量的离散化是通过模糊集合的隶属度函数将精确值模糊化的。对于论域为离散，且元素个数为有限时，模糊集合的隶属度函数可以用向量或者表格的形式来表示。如把在[-4，+4]之间变化的连续量分为 7 个标记，每一个标记对应一个模糊集，这样处理使模糊化过程比较简单。否则，将每一精确量对应一个模糊子集，有无穷多个模糊子集，使模糊化过程复杂化。必须强调指出的是，实际上的输入变量（如误差和误差的变化等）都是连续变化的量，通过模糊化处理，把连续量离散为[-4，+4]之间有限个整数值的做法是为了使模糊推理合成的方便。此时，采用数值方法用表格描述隶属度的例子如表 8.1 所示。表中每一行表示一个模糊集合的隶属度函数。例如：

$$ZO=\frac{0.5}{-1}+\frac{1}{0}+\frac{0.5}{1}$$

表 8.1 隶属函数的离散化表格

模糊量＼论域	-4	-3	-2	-1	0	1	2	3	4
NB	1	0.5	0	0	0	0	0	0	0
NS	0	0.5	1	0.5	0	0	0	0	0
ZO	0	0	0	0.5	1	0.5	0	0	0
PS	0	0	0	0	0	0.5	1	0.5	0
PB	0	0	0	0	0	0	0	0.5	1

对于论域为连续的情况，隶属度常常采用函数的形式来描述，最常见的为三角形函数和指数形函数。如指数形函数的解析式为：

$$\mu_A(x)=\mathrm{e}^{-\frac{(x-x_0)^2}{2\sigma^2}}$$

其中，x_0是隶属度函数的中心值，σ^2是方差。

2）单值模糊化方法

单值模糊化方法是将在某区间的一个精确量转换成确定论域中的模糊单值，使其仅在该点处的隶属度为 1，而其他各点隶属函数均为 0。单值模糊化通常又称为取“棒形”或“I 形”隶属函数。本质上，模糊单值仍是一个确定值；因此，在这种模糊化过程中并没有引入模糊性，然而在模糊控制应用中，这种方法由于使人感到自然和易于实现而得到了广泛应用。

8.5.2 解模糊判决方法

如上所述，经过模糊推理得到的控制输出是一个模糊隶属函数或者模糊子集，它反映了控制语言的模糊性质，这是一种不同取值的组合。然而在实际应用中要控制一个物理对象，只能在某一个时刻有一个确定的控制量，这就必须要从模糊输出隶属函数中找出一个最能代表这个模糊集合及模糊控制作用可能性分布的精确量，这就是解模糊（Defuzzification）。从数学上讲，这是一个从输出论域所定义的模糊控制作用空间到精确控制作用空间的映射。解模糊可以采用不同的方法，用不同的方法所得到的结果也是不同的。理论上用质心法比较合理，但是计算比较复杂，故在实时性要求高的系统中有时不采用这种方法。最简单的方法是最大隶属度方法，这种方法取所有模糊集合或者隶属函数中隶属度最大的那个值作为输出，但是这种方法未顾及其他隶属度较小的那些值的影响，代表性不好，所以它经常用于简单的系统。介于这两者之间的还有各种平均法：如加权平均法、隶属度限幅（α-cut）元素平均法等。下面以“水温适中”为例，说明不同方法的计算过程。

这里假设“水温适中”的隶属度函数为：

$$\mu_N(x_i) = \{X:\ 0.0/0 + 0.0/10 + .033/20 + 0.67/30 + 1.0/40 + 1.0/50$$

$$+ 0.75/60 + 0.5/70 + 0.25/80 + 0.0/90 + 0.0/100\}$$

1）质心法

所谓质心法，就是取模糊隶属度函数曲线与横坐标轴围成面积的质量中心作为代表点。理论上说，我们应该计算输出范围内一系列连续点的重心，即：

$$u = \frac{\int_x \mu_N(x)x\mathrm{d}x}{\int \mu_N(x)\mathrm{d}x} \tag{8.2}$$

但实际上我们是通过计算输出范围内整个采样点（即若干离散值）的质心。这样在不花太多时间的情况下，用足够小的取样间隔来提供所需要的精度，这是一种最好的折衷方案。即：

$$
\begin{aligned}
u &= \frac{\sum x_i \cdot \mu_N(x)}{\sum \mu_N(x)} \\
&= (0.0 \cdot 0 + 0.0 \cdot 10 + .033 \cdot 20 + 0.67 \cdot 30 + 1.0 \cdot 40 + 1.0 \cdot 50 \\
&\quad + 0.75 \cdot 60 + 0.5 \cdot 70 + 0.25 \cdot 80 + 0.0 \cdot 90 + 0.0 \cdot 100) \\
&\quad /(0.0 + 0.0 + 0.33 + 0.67 + 1.0 + 1.0 + 0.75 + 0.5 + 0.25 + 0.0 + 0.0) \\
&= 48.2
\end{aligned}
\tag{8.3}
$$

2） 最大隶属度法

这种方法最简单，只要在推理结论的模糊集合中取隶属度最大的那个元素作为输出量即可。不过要求这种情况下其隶属度函数曲线一定是正规凸模糊集合（即其曲线只能是单峰曲线）。如果该曲线是梯形平顶的，那么具有最大隶属度的元素就可能不止一个，这时就要对所有取最大隶属度的元素求平均值。

例如对于“水温适中”，按最大隶属度原则，有两个元素 40 和 50 具有最大隶属度 1.0，那就要对所有取最大隶属度的元素 40 和 50 求平均值，执行量应取：

$$u_{\max} = (40 + 50) / 2 = 45 \tag{8.4}$$

该方法简便易行，实时性也好，但它概括信息量少，因为它完全不考虑其余一切从属程度较小点的情况。

3） 系数加权平均法

这种方法就是依照普通加权平均公式，按下式来计算控制量：

$$u = \frac{\sum k_i x_i}{\sum k_i} \tag{8.5}$$

这里的系数 k_i 的选择要根据实际情况而定，不同的系数就决定系统有不同的响应特性。当该系数选择 $k_i = \mu_N(x_i)$ 时，即取其隶属函数时，这就是质心法。在模糊逻辑控制中，可以通过选择和调整该系数来改善系统的响应特性，这种方法具有灵活性。

4） 隶属度限幅元素平均法

用所确定的隶属度值 α 对隶属函数曲线进行切割，再对切割后等于该隶属度的所有元素进行平均，用这个平均值作为输出执行量，这种方法就称为隶属度限幅（α-cut）元素平均法。

例如，当取 α 为最大隶属度值时，表示“完全隶属”关系，这时 $\alpha = 1.0$。在“水温适中”的情况下，40℃和 50℃的隶属度是 1.0，求其平均值得到输出代表量：

$$u = (40 + 50) / 2 = 45$$

当 $\alpha = 0.5$ 时，表示“大概隶属”关系，切割隶属度函数曲线后，这时假定从 30℃到 70℃的隶属度值都包含在其中，所以求其平均值得到输出代表量：

$$u = (30 + 40 + 50 + 60 + 70) / 5 = 50$$

另外，单值法也常用于解模糊中。

8.6 模糊控制规则

模糊控制规则是设计模糊控制器的核心，因此如何建立模糊规则就成为一个十分关键的问题。模糊控制器控制规则的建立应该是人们在手动控制过程中经过长期实践，不断修正完善的一套行之有效的控制策略。模糊控制器的最大特点就是只要有了有经验的操作者或专家给出的控制规则，就可以根据所选定的模糊推理方法获得合理的控制量。一般情况下是通过以下几种方式来获得模糊控制规则的。

1）基于专家的经验和控制工程知识

模糊控制规则具有模糊条件句的形式，它建立了前件中的状态变量与后件中的控制变量之间的联系。通过总结人类专家的经验，并用适当的语言来加以表述，最终可表示成模糊控制规则的形式，或通过获得特定应用模糊控制规则的原理，再经一定的试凑和调整，可获得具有更好性能的控制规则。

2）基于操作人员的实际控制过程

在许多人工控制的工业系统中，很难建立控制对象的模型，因此采用常规的控制方法来对其进行设计和仿真比较困难。然而熟练的操作人员却能成功地控制这样的系统。事实上，操作人员有意无意地使用了一组 IF-THEN 模糊规则来进行控制。但是他们往往并没有语言明确地将它们表达出来，因此可以通过记录操作人员实际控制过程时的输入/输出数据，从中总结出模糊控制规则。

3）解析控制规则公式

在经验所获规则不全面或根本没有经验规则的情况下，可以采用以下的解析控制规则公式。

（1）简单的解析控制规则

在简单的模糊控制规则的算法中，可将误差 e、误差变化率 ec 和控制量 u 之间的控制规则表示为如下的解析公式：

$$u = -\mathrm{int}\frac{e+ec}{2} \tag{8.6}$$

其中，int 表示取整。

采用解析表达式描述的控制规则简单方便，使得输入和输出之间的关系式可以直接进行计算，更易于计算机的实时控制。

（2）带有一个可调因子的控制规则

从模糊控制规则的解析表达式(8.6)中可见，控制作用的大小仅取决于误差和误差的变化，并且对两者采取同样的重视程度。为了适应不同的被控对象的控制性能要求，在(8.7)式的基础上引入一个调节因子α，则可得到一种带有一个调整因子的控制规则：

$$u = -\mathrm{int}[e + (1-\alpha)ec] \tag{8.7}$$

改变α的大小，意味着对误差和误差的变化不同的重视程度。实验结果表明，当被控对象阶次较低时，应取$\alpha > 0.5$；相反，当被控对象阶次较高时，应取$\alpha < 0.5$；当取$\alpha = 0.5$时，(8.7)式就变成(8.6)式了。

（3）带有多个可调因子的控制规则

如果对不同的误差等级引入不同的调整因子的值，就可构成带有多个可调因子的控制规则，这种设计方式有利于满足控制系统在不同被控状态下对调整因子的不同要求。例如，当误差 e、误差变化率 ec 及控制量 u 的论域取为：

$$\{e\}=\{ec\}=\{u\}=\{-3,-2,-1,0,1,2,3\}$$

则带有多个可调因子的控制规则可表示为：

$$u=\begin{cases}-\mathrm{int}[\alpha_0 e+(1-\alpha_0)ec], & e=0\\ -\mathrm{int}[\alpha_1 e+(1-\alpha_1)ec], & e=\pm 1\\ -\mathrm{int}[\alpha_2 e+(1-\alpha_2)ec], & e=\pm 2\\ -\mathrm{int}[\alpha 3 e+(1-\alpha_3)ec], & e=\pm 3\end{cases} \tag{8.8}$$

其中调整因子 α_0、α_1、α_2、$\alpha_3\in(0,1)$。

4)基于学习

许多模糊控制主要是用来模仿人的决策行为，但很少能够根据经验和知识产生模糊控制规则并对它们进行修改的能力。随着模糊自组织控制以及神经模糊系统的出现，使得模糊控制器具有自身学习能力，并具备了通过学习产生合适的控制规则的能力。

8.7 模糊逻辑推理

模糊推理是设计模糊控制器的关键。常用的模糊控制规则的推理方法有三种：合成模糊推理方法、结论是线性函数的模糊推理方法和合成推理的解析公式法。

8.7.1 合成模糊推理法

实际应用中使用较广泛的合成模糊推理法有以下两种。

1）直接推理法

由量化论域中的各输入量、输出量求出每条控制规则的模糊关系 R_{ij}，再使用模糊关系 R_{ij} 计算出各输出控制分量，最后算出总输出控制量。直接推理法的特点是不用计算出总的模糊关系 R。

2）间接推理法

由量化论域中的各输入量、输出量及控制规则求出总的模糊关系 R，再使用总的模糊关系 R 计算出输出控制量。

下面给出两种合成模糊推理法的计算公式。

假设有两个输入变量 E 和 EC、一个输出变量 U 以及一个控制规则表，其中，E 的模糊标记为 $A_1,A_2,\ldots,A_m$；EC 的模糊标记为 $B_1,B_2,\ldots,B_n$；U 的模糊标记为 $C_1,C_2,\ldots,C_l$；控制规则有 $m\times n$ 个，其格式为：

$$\text{if} \quad A_i \quad \text{and} \quad B_j \quad \text{then} \quad C_{ij}$$

其中，A_i 属于论域 $\{A_1, A_2, \ldots, A_m\}$

B_j 属于论域 $\{B_1, B_2, \ldots, B_n\}$

C_{ij} 属于论域 $\{C_1, C_2, \ldots, C_l\}$

现有输入 A^*，B^*，需求出 C^*。

3）直接法合成推理公式

$$R_{ij} = A_i \times B_j \times C_{ij} = (A_i \times B_j) \circ C_{ij}$$

$$C_{ij}^* = (A^* \times B^*)^T \circ R_{ij}$$

$$C^* = \bigvee_{i=1, j=1}^{m,n} C_{ij}^*$$

4）间接法合成推理公式

$$R_{ij} = A_i \times B_j \times C_{ij} = (A_i \times B_j) \circ C_{ij}$$

$$R = \bigvee_{i=1, j=1}^{m,n} R_{ij}$$

$$C^* = (A^* \times B^*)^T \circ R$$

实际上，无论采用直接法还是间接法进行模糊逻辑推理，所求出的控制值都是相同的。欲证直接法与间接法的模糊逻辑推理结果一致，只需证明下面的等式成立：

$$(A \times B) \circ \bigvee_{i=1, j=1}^{m,n} R_{ij} = \bigvee_{i=1, j=1}^{m,n} (A \times B) \circ R_{ij}$$

合成运算"∘"采用最大—最小法的证明如下：

$$\begin{aligned}
\mu_C(z) &= (A \times B) \circ \bigvee_{i=1, j=1}^{m,n} R_{ij} \\
&= [\mu_A(x) \wedge \mu_B(y)] \circ \bigvee_{i=1, j=1}^{m,n} [\mu_{R_{ij}}(x, y, z)] \\
&= \bigvee_{x,y} \{[\mu_A(x) \wedge \mu_B(y)] \wedge [\bigvee_{i=1, j=1}^{m,n} (\mu_{R_{ij}}(x, y, z))]\} \\
&= \bigvee_{x,y} \bigvee_{i=1, j=1}^{m,n} \{[\mu_A(x) \wedge \mu_B(y)] \wedge \mu_{R_{ij}}(x, y, z)\} \\
&= \bigvee_{i=1, j=1}^{m,n} \{[\mu_A(x) \wedge \mu_B(y)] \circ \mu_{R_{ij}}(x, y, z)\} \\
&= \bigvee_{i=1, j=1}^{m,n} (A \times B) \circ R_{ij}
\end{aligned}$$

对于"∘"采用最大—积合成法的情形，只要把上式中的取小运算换为乘积运算，同样可以证出相同的结果。虽然采用直接法和间接法进行模糊逻辑推理所获结果相同，但从实际的计算机操作中证实两者所用运算时间不同，间接法所用计算时间较少。

由上可知，实现上述推理过程必须执行复杂的矩阵运算，计算工作量太大。因此，在

线推理时很难满足控制系统的实时性要求。正因为如此，在模糊控制器的设计中，最常使用的是一种所谓的“查询表”，这种控制器是通过模糊控制规则表、模糊化的输入量以及合成推理法，计算出输入/输出之间关系的对应表格，或称为“控制表”，由计算机事先离线计算好，存储在内存中供实时查表使用。实际应用时，先根据量化后的输入误差及误差变化值，直接从控制表中查表求出对应的输出控制量，再乘以比例因子，即可作为输出去控制被控对象。以此可以大大提高模糊控制的实施效果，节省内存空间。

下面采用间接的合成推理法，重新对例 7.1，对模糊化采用把输入/输出连续量离散化为论域上的模糊子集进行离线制作控制表的步骤。

（1） 模糊化

①输入/控制量的模糊语言描述

描述输入/控制量的语言值模糊子集选取为：

$$\{NB, NS, ZO, PS, PB\}$$

其中：NB = 负大；NS = 负小；ZO = 零；PS = 正小；PB = 正大。

②输入/控制量的量化

设温度的论域为 X，将其论域变化范围[0，40]量化为 9 个等级，量化因子 $K_x = \dfrac{9}{40}$，则有：

$$X = \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$$

设湿度的论域为 Y，将其论域变化范围[0%，30%]也量化为 9 个等级，量化因子为 $K_y = \dfrac{9}{30}$，则有：

$$Y = \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$$

设控制量浇水时间的论域为 U，将其论域变化范围[0，70]同样量化为 9 个等级，量化因子为 $K_u = \dfrac{9}{70}$，则有：

$$Z = \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$$

③输入/控制量隶属函数的建立

选用等距离三角形作为输入/控制量的隶属函数，如图 8.11 所示。

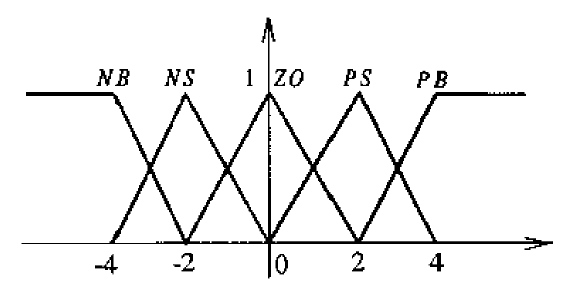


图 8.11 输入/控制量的隶属函数

④模糊化表的建立（X，Y，Z）

为了便于制表，将输入、输出变量值通过论域离散化，用有限个数量值表示出其与隶属函数之间的关系式，即通过作表法来完成模糊化过程。本例中的两个输入 X 和 Y 以及

输出 Z 的模糊化表具有相同的形式，如表 8.2 所示。

表 8.2 隶属函数的离散化表格

论域 模糊量	-4	-3	-2	-1	0	1	2	3	4
NB	1	0.5	0	0	0	0	0	0	0
NS	0	0.5	1	0.5	0	0	0	0	0
ZO	0	0	0	0.5	1	0.5	0	0	0
PS	0	0	0	0	0	0.5	1	0.5	0
PB	0	0	0	0	0	0	0	0.5	1

（2）模糊控制规则表的建立

模糊控制规则表如表 8.3 所示。

表 8.3 模糊控制规则表

X Z Y	*NB*	*NS*	*ZO*	*PS*	*PB*
NB	*PB*	*PB*	*PS*	*PS*	*ZO*
NS	*PB*	*PS*	*PS*	*ZO*	*ZO*
ZO	*PS*	*PS*	*ZO*	*ZO*	*NS*
PS	*PS*	*ZO*	*ZO*	*NS*	*NS*
PB	*ZO*	*ZO*	*NS*	*NS*	*NB*

①控制表格的建立（采用间接推理法）；

②求 R_{ij}。

a) 由模糊化表可知：

$$A_1 = B_1 = C_1 = NB = (1,0.5,0,0,0,0,0,0,0)$$

$$A_2 = B_2 = C_2 = NS = (0,0.5,1,0.5,0,0,0,0,0)$$

$$A_3 = B_3 = C_3 = ZO = (0,0,0,0.5,1,0.5,0,0,0)$$

$$A_4 = B_4 = C_4 = PS = (0,0,0,0,0,0.5,1,0.5,0)$$

$$A_5 = B_5 = C_5 = PB = (0,0,0,0,0,0,0,0.5,1)$$

b) 由模糊控制规则

如果 X 是 A_1，且 Y 是 B_1，那么 Z 是 C_{11}；

即

如果 X 是 NB，且 Y 是 NB，那么 U 是 PB；

结合模糊控制规则表，可得：

$C_{11}=C_{12}=C_{21}=PB$

$C_{13}=C_{14}=C_{22}=C_{23}=C_{31}=C_{32}=C_{41}=PS$

$C_{15}=C_{24}=C_{25}=C_{33}=C_{34}=C_{42}=C_{43}=C_{51}=C_{52}=ZO$

$C_{35}=C_{44}=C_{45}=C_{53}=C_{54}=NS$

$C_{55}=NB$

$R_{ij}=A_i\times B_j\times C_{ij}=(A_i\times B_j)^{T1}\circ C_{ij}$

其中 $(A_i\times B_j)^{T1}$ 表示将 $A_i\times B_j$ 按行拉直成行向量，再转置成列向量。

c）求模糊关系矩阵 R

$$R=\bigcup_{i=1,j=1}^{i=5,j=5}R_{ij}$$

通过计算机计算，可得模糊关系矩阵 $R_{81\times 9}$。

（3）计算控制量 C^* 的矩阵

$$C^*=(A^*\times B^*)^{T2}\circ R$$

其中的 $(A^*\times B^*)^{T2}$ 表示将 $A^*\times B^*$ 按行拉直成行向量。A^* 和 B^* 为分别取{NB，NS，ZO，PS，PB}中不同的模糊量值，C^* 为所对应的值。

将上述推理过程编成程序，通过计算机求解后可得对于输入变量论域{-4，-3，-2，-1，0，1，2，3，4}中的每一组输入，所对应的控制器的输出 $C_{ij}^*(i,j=1,2,\ldots,9)$ 的隶属度。如：

$C_{11}^*=$（0,0,0,0,0,0,0,0.5,1）；

$C_{12}^*=$（0,0,0,0,0,0,0,0.5,0.5）；

…

$C_{21}^*=$(0,0,0,0.5,1,0.5,0,0,0)；

$C_{22}^*=$(0,0,0,0,0,0,0,0.5,0.5) ；

…

$C_{98}^*=$(0.5,0.5,0.5,0.5,0,0,0,0,0)；

$C_{99}^*=$（1,0.5,0,0,0,0,0,0,0）。

（4）解模糊

根据公式 $C=\dfrac{\sum\mu\cdot Z}{\sum\mu}$，通过编程由计算机分别计算后可得到对输入论域中不同离散值

的输出控制表如表 8.4 所示。

表 8.4 模糊控制器的控制表

Z \ X / Y	-4	-3	-3	-1	0	1	2	3	4
-4	3.667	3.5	3.667	2.5	2	2	2	1	0
-3	3.5	2.5	2.5	2.5	2	1	1	1	0
-2	3.667	2.5	2	2	2	1	0	0	0
-1	2.5	2.5	2	1	1	1	0	-1	-1
0	2	2	2	1	0	0	0	-1	-2
1	2	1	1	1	0	-1	-1	-1	-2
2	2	1	0	0	0	-1	-2	-2	-2
3	1	1	0	-1	-1	-1	-2	-2.5	-2.5
4	0	0	0	-1	-2	-2	-2	-2.5	3.667

很显然，这种制表法不同于我们例 7.1 中所采用的方法，主要的不相同点在于制表法中采用的是将输入及输出变量在整个论域上的连续量模糊化为有限个数的量化值，而例 7.1 中输入/输出变量的模糊化采用的是直接通过模糊隶属函数进行单值模糊化，为了说明问题的方便起见，这里以三角形隶属函数为例来说明其输出分量的求解过程。

先考虑只有一条控制规则的情况。设有模糊推理语句：

R_{11}： 如果 x 是 A_1 and y 是 B_1，那么 z 是 C_{11}

其中 $A_1 \in \{a1, a2, a3\}$，$B_1 \in \{b1, b2, b3\}$，$C_{11} \in \{c1, c2, c3\}$，$a1, a2, a3, b1, b2, b3, c1, c2, c3$ 分别表示变量 x，y，z 在各自论域中离散点处的隶属度。

①求此规则的蕴含关系 $R_{11} = (A_1 \text{ and } B_1) \to C_{11}$

若对于 A_1 and B_1 采用求交运算，蕴含关系采用最小运算，可得：

$$A_1 \text{ and } B_1 = A_1 \times B_1 = A_1^T \wedge B_1$$

$$= \begin{bmatrix} a1 \\ a2 \\ a3 \end{bmatrix} \wedge \begin{bmatrix} b1 & b2 & b3 \end{bmatrix}$$

$$= \begin{bmatrix} a1 \wedge b1 & a1 \wedge b2 & a1 \wedge b3 \\ a2 \wedge b1 & a2 \wedge b2 & a2 \wedge b3 \\ a3 \wedge b1 & a3 \wedge b2 & a3 \wedge b3 \end{bmatrix}$$

为了便于下面进一步的计算，可将 $A_1 \times B_1$ 的模糊矩阵表示成如下的拉直行向量的转置：

$$R_{A_1\times B_1}=(A_1\times B_1)^{T1}$$
$$=[a1\wedge b1\quad a1\wedge b2\quad a1\wedge b3\quad a2\wedge b1\quad a2\wedge b2\quad a2\wedge b3\quad a3\wedge b1\quad a3\wedge b2\quad a3\wedge b3]^T$$

从而蕴含关系：

$$R_{11}=(A_1 \text{ and } B_1)\to C_{11}=(A_1\times B_1)^{T1}\wedge C_{11}$$

$$=\begin{bmatrix} a1\wedge b1\wedge c1 & a1\wedge b1\wedge c2 & a1\wedge b1\wedge c3\\ a1\wedge b2\wedge c1 & a1\wedge b2\wedge c2 & a1\wedge b2\wedge c3\\ \cdots\cdots & & \\ a3\wedge b3\wedge c1 & a3\wedge b3\wedge c2 & a3\wedge b3\wedge c3 \end{bmatrix}$$

②计算输入量的模糊集合 A' and B'

当有输入 x'和 y'激活了模糊控制规则 R_{11}，由于此时采用单值模糊化，可得 $x'=A_1'$，$y'=B_1'$为：

$$A_1' = (1,0,0),\quad B_1'=(1,0,0)$$

则有

$$A_1'\times B_1'=\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix}$$

同样为了后面合成的方便，需将 $A_1'\times B_1'$拉直成行向量，

③计算输出分量 C_{11}'

根据最大—最小合成推理方法，对应的输出 C_{11}'为

$$C_{11}'=(A_1'\times B_1')\circ R_{11}$$
$$=(1,0,0,0,0,0,0,0,0)\circ R_{11}$$
$$=(a1\wedge b1\wedge c1, a1\wedge b1\wedge c2, a1\wedge b1\wedge c3)$$

或者，对于同一输入 $x'=A_1'$，$y'=B_1'$，有可能会有

$$A_1'=(1,0,0),\quad B_1'=(0,1,0)$$

此时

$$A_1'\times B_1'=\begin{bmatrix}0&1&0\\0&0&0\\0&0&0\end{bmatrix}$$

同理，需将 $A_1'\times B_1'$拉直成行向量，然后根据最大—最小合成推理方法，对应的输出 C_{11}'为：

$$
\begin{aligned}
C_{11}' &= (A_1' \times B_1') \circ R_{11} \\
&= (0,1,0,0,0,0,0,0,0) \circ R_{11} \\
&= (a1 \wedge b2 \wedge c1, a1 \wedge b2 \wedge c2, a1 \wedge b2 \wedge c3)
\end{aligned}
$$

以此类推，不难发现对于 A_1' 的论域和 B_1' 的论域，不管它们的元素有多少，当 A_1' 量化之后只会选中某一个元素，例如第 i 个元素，并且具有该元素的隶属度为 1，其余均为 0，即有：

$$A_1' = (\ 0, \ldots, 0, \underset{\text{第 } i \text{ 个}}{1}, 0, \ldots, 0\ )$$

同理，B_1' 量化后也会只选中某一个元素，例如第 j 个元素，且只有该元素的隶属度为 1，其余均为 0，即有：

$$B_1' = (\ 0, \ldots, 0, \underset{\text{第 } j \text{ 个}}{1}, 0, \ldots, 0\ )$$

此时不难得到 C_{11}' 为：

$$\mu_{C_{11}'} = \mu_{A_1}(x) \wedge \mu_{B_1}(y) \wedge \mu_{C_{11}}(z)$$

当有输入 x' 和 y'，可能激活其他模糊控制规则，如 R_{12}：

R_{12}：　如果 x 是 A_1 and y 是 B_2，那么 z 是 C_{12}

同理，可求得其对应输出为：

$$\mu_{C_{12}'} = \mu_{A_1}(x) \wedge \mu_{B_2}(y) \wedge \mu_{C_{12}}(z)$$

由求解 $\mu_{C_{11}'}$ 和 $\mu_{C_{12}'}$ 的算式表明，对于输入 x' 和 y'，它所激活的每一条模糊控制规则的输出分量等于用输入变量 x，y 对隶属函数的模糊集合上所获得的隶属度进行相互结合，分别取其中较小的值作为总的模糊推理前提的隶属度，再以此为基准去切割推理结论的隶属函数，结论 $\mu_{C_{ij}'}$ 等于 $\mu_{C_{ij}}$ 被激活强度箝位后的结果。输入为单值模糊化，推理采用最大—最小合成法时的图形解释如图 8.12 所示。

根据以上每一条模糊控制规则输出分量的计算公式可以看出，当输入采用单值模糊化时，所对应的输出模糊量的推理过程将大大简化，对于给定的输入变量 x 和 y，只要在输入变量的论域与隶属函数的关系式（图）中分别求出与之对应的所有模糊标记上的隶属度，然后对隶属函数的模糊集合上所获得的隶属度进行两两结合，构成由输入变量所激活的模糊控制规则。另一方面，从设计隶属函数的角度我们已知道，通常对于同一个论域值，所相交的隶属函数的数目不多于两个，所以对于输入变量 x 和 y，所激活的模糊控制规则的数目一般不会多于 4（=2*2）个，所以这种模糊推理运算法极大地简化了运算过程，并且对于输入量不需要进行量化处理，可以直接进行单值模糊化，从这点上讲，可以提高推理精度。在实际应用的实时计算处理中通常均采用此种操作。

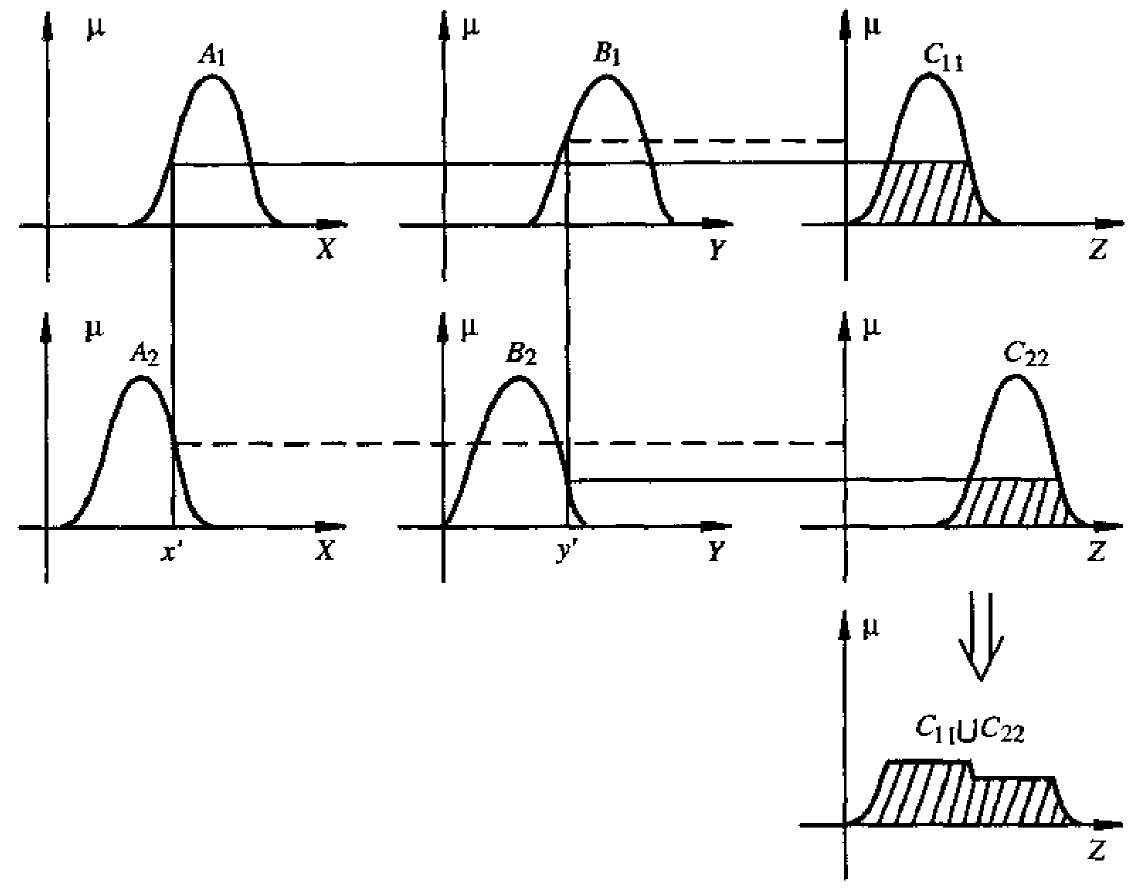


图 8.12 输入为模糊单值且采用最大一最小进行模糊推理时的图形解释

下面给出输出变量 C' 完整的模糊推理求解过程。

①对于每一组精确的输入变量值（x,y）通过模糊化隶属函数图，求得输入值对应在各模糊标记上的隶属度 $\alpha_1=\mu_{A_1}(x)$， $\beta_1=\mu_{B_1}(y)$， $\alpha_2=\mu_{A_2}(x)$ 和 $\beta_2=\mu_{B_2}(y)$。由此激活控制表中与之发生作用的控制规则（一般只有 4 条），例如激活了：

$$C(\mu_{A_1}(x),\mu_{B_1}(y)),\ C(\mu_{A_1}(x),\mu_{B2}(y)),\ C(\mu_{A_2}(x),\mu_{B_1}(y)),\ C(\mu_{A_2}(x),\mu_{B_2}(y))$$

②通过对隶属度的最小运算获得所对应的规则的隶属度

隶属度	控制规则
$(\alpha_1\wedge\beta_1)$	C_{α_1,β_1}
$(\alpha_1\wedge\beta_2)$	C_{α_1,β_2}
$(\alpha_2\wedge\beta_1)$	C_{α_2,β_1}
$(\alpha_2\wedge\beta_2)$	C_{α_2,β_2}

③利用取最大值的方法求出输出变量的隶属度，即如果 C_{α_1,β_1}， C_{α_1,β_2}， C_{α_2,β_1} 和

C_{α_2,β_2} 中的模糊标记有重复的，则从中取隶属度较大的值。

④解模糊。通过同样的解模糊法（例如质心法）求出其精确值。

此方法的优点是由于不需要对每一组输入值在所有的 $m\times n$ 个规则里进行模糊推理操作，其计算量远远小于上述制表法中所采用的方法，该方法很适合用于在线实时计算控制输出量，并且在实际计算时不需要对输入进行量化处理，直接由精确输入值所对应的各模糊标记的隶属度求出对应的输出。

8.7.2 结论是线性函数的模糊推理方法

这种模糊推理法的特点是推理的结论部分是前提语言变量值的函数，又称为 Sugeno 推理法。一般形式为：

$$\text{if } x1 = A1 \text{ and } x2 = A2 \text{ and } \cdots xn = An \text{ and then } y = f(x1, x2, \cdots, xn)$$

结论函数的简单形式可表示为：

$$\text{if } x1 = A1 \text{ and } x2 = A2 \text{ and } \cdots xn = An \text{ and then } y = p0 + p1x1 + \cdots + pnxn$$

在上面条件语句中，$x1, x2, \ldots, xn$ 是语言变量，$A1, A2, \ldots, An$ 是语言变量值，$p0, p1, \ldots, pn$ 是系数。

对于有 k 条规则的模糊控制，则可表示为：

R_i if $x1_i = A1_i$ and $x2_i = A2_i$ and$\cdots$ and $xn_i = An_i$ then $y_i = p0_i + p1_i x1_i + \cdots + pn_i xn_i$ 式中，$i = 1, 2, \ldots, k$。

设有输入 $x1^*, x2^*, \ldots xn^*$，则输出控制量的精确值 y 的计算过程如下：

①对每条规则 R_i，计算其结论 y_i 的值。

$$y_i = p0_i + p1_i x1_i + p2_i x2_i + \cdots + pn_i xn_i$$

式中，$i = 1$，2，…，k。

②求命题 $y = \ y_i$ 的真值，即结论的权值。

命题 $y = y_i$ 的真值用 $|w_i|$ 表示，且有

$$|w_i| = \mathop{\wedge}_{j=1}^{n} \mu_{Aji}(xj_i)$$

③求输出控制的精确值。若采用“质心法”求精确值，则有：

$$y = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i}$$

计算式中的 $w_i = |y - y_i|$。

例 8.1 设有 3 条控制规则，且分别为：

R_1 : if $x1_1 = S_1$ and $x2_2 = S_2$ then $y = x1_1 + x2_2$

R_2 : if $x1_1 = B_1$ then $y = 2x1_1$

R_3 :if $x1_1 = B_2$ then $y = 3x1_2$

式中，S_1 表示小 1；S_2 表示小 2；B_1 表示大 1；B_2 表示大 2。

当输入 $x1=12, x2=5$ 时，求输出值 y。

先求每条规则结论 y_i 的值。且有：

$$y_1 = x1 + x2 = 17$$
$$y_2 = 2x1 = 24$$
$$y_3 = 3x2 = 15$$

然后求每条规则的权重，即有：

$$w_1 = \mu_{S_1}(x1) \wedge \mu_{S_2}(x2) = \mu_{S_1}(12) \wedge \mu_{S_2}(5)$$
$$w_2 = \mu_{B_1}(x1) = \mu_{B_1}(12)$$
$$w_3 = \mu_{B_2}(x2) = \mu_{B_2}(5)$$

设有：

$$\mu_{S_1}(12) = 0.25, \mu_{S_2}(5) = 0.375, \mu_{B_1}(12) = 0.2, \mu_{B_2}(5) = 0.375$$

则有：

$$w_1 = 0.25, w_2 = 0.2; w_3 = 0.375$$

因此，有精确值 y：

$$y = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i} = 17.8$$

当只有一个规则时，上式中 $w_2 = w_3 = 0$，那么，这就相当于线性控制。当有多个规则时，输入空间可分为若干模糊子空间，在每个子空间找出线性的输入/输出关系，这些子空间的集合与整个的非线性输入/输出关系可以看成是等价的。在这里因为只有前提部分是用语言来描述的，所以，它不像第一种模糊推理法中的控制规则那样容易理解，但是实际上它的描述能力也许是最好的，因为它用少数的语言规则可生成较复杂的非线性函数。

8.7.3 量化因子及比例因子的选择

在实现模糊控制算法时，通过每隔一定时间（采样周期）采样被控对象的输出信号（数字量）后，把该数字量和内部设定的数字信号（参考输入信号）进行比较就可以得到当前控制用的输入变量信号（误差信号）。前后两次采样对应的误差信号除以时间间隔就是误差变化率信号。为了进行模糊运算，必须把这两个精确量转换为模糊集的论域的某一个相应的值。这实际上就是要进行基本论域（精确量）到模糊集的论域（模糊量）的转换，这种转换过程的实现就需要引入量化因子的概念。

量化因子一般用 K 表示，误差的量化因子 K_e、误差变化率的量化因子 K_{ec} 和控制量的比例因子 K_u 分别由以下计算公式确定：

$$K_e=\frac{n}{x_e}, \qquad K_{ec}=\frac{m}{x_{ec}}, \qquad K_u=\frac{y_u}{l}$$

设计模糊控制器除了要有一整套有效的控制规则外，还必须合理地选择模糊控制器的量化因子和比例因子系数。大量实验结果表明，量化因子和比例因子的大小及两个量化因子之间的大小相对关系，对模糊控制器的控制性能影响极大，并且需要通过整个控制系统的调试（可先通过仿真）来完成。

1）K_e、K_{ec}对系统动态性的影响

一般说来，K_e、K_{ec} 变化时，实际误差和误差变化所对应的论域上的原始值也将发生变化。K_e、K_{ec} 越大，对应的语言值也越大，反之亦然。在误差变化所取语言值不变的条件下，误差所取语言值越大，相应控制器的输出（控制量变化）所取语言值也越大；而在误差所取语言值不变的条件下，误差变化所取语言值越大，相应的控制器的输出所取语言值越小。

K_e 对动态特性的影响是：K_e 大，调节死区小，上升速率大。但是，K_e 取的过大，将使系统产生较大的超调，调节时间增大，甚至产生震荡，使系统不能稳定工作。

K_{ec} 对动态性的影响是：K_{ec} 大，反应较迟钝；K_{ec} 小，反应快，上升速率大。而 K_{ec} 过小，引起大的超调，使调节时间长，严重时不能稳定工作。

2）K_e、K_{ec}对系统稳定性的影响

在模糊控制系统中，一般不可能消除稳态误差，更不可能消除误差变化率。一般而言，K_e 增加，稳态误差将减小；K_{ec} 增大，稳态时误差变化率也将减小。然而 K_e、K_{ec} 对动态性能也有影响，因此必须兼顾两方面的性能。

3）K_u对系统性能的影响

K_u 相当于常规系统中的比例增益，它主要影响控制系统的动态性能。一般 K_u 加大，上升速度就快。但 K_u 过大，将产生较大的超调，严重时会影响稳态工作，和一般控制系统不同的是，K_u 一般不影响系统的稳态误差。

第9章 运动控制中的摩擦力补偿及其建模技术

9.1 引 言

摩擦力最早是由Da Vinci于1519年发现的，他认为，摩擦力正比于负载，并与运动方向相反，且与物体接触面积无关。Da Vinci的摩擦力模型是由Amontons于1699年发现的。Morin于1833年引入了静摩擦力的概念，而Reynolds于1886年提出了流体的粘滞摩擦力，从而完成了最常用于工程的摩擦力模型：静态+库仑+粘性摩擦力的模型。

摩擦存在于定位控制、低速和速度变向的跟踪系统中，包括望远镜、天线、精密机床、磁带驱动器、机器人和跟踪机械的控制系统。摩擦学于20世纪30年代诞生于英国。近70年来，众多研究者的研究成果也提供了一系列的摩擦力模型以及对其的补偿技术。人们一直认为：摩擦补偿器的成功设计与分析，极大地取决于所采用的摩擦模型的精度以及应用的分析技术的合理性。摩擦学研究者从接触摩擦学角度，通过对一致性以及不一致性接触表面的物理特性的研究，分析了刚体接触面的粗糙度、接触应力、剪切强度、负载变化等因素，对摩擦力的影响作用，相当透彻地研究了具有润滑的金属与金属表面的接触，在慢速滑动过程中，作为速度函数的摩擦力，在摩擦机理上所体现出来的四个动力学区域，并通过许多分析工具诸如：描述函数法、相平面法、代数分析法及仿真，加上细致周到的实验，求出或经验地得到比较精确的摩擦力模型。此外，机械工程学者也从不同角度去考虑具有摩擦的机器的补偿技术问题。其中包括机械学、任务本身、摩擦力模型、分析技术以及其他补偿技术，其方法主要集中在低摩擦机器的设计或润滑的选择、带有死区的积分控制、直接力反馈、脉冲控制、库仑摩擦力的前向补偿以及取决于位移的摩擦力的前向补偿等。

许多控制领域的研究者，利用稳定性理论、非线性控制、非线性系统辨识、自适应控制以及其他手段，对具有摩擦力的机电系统进行了不同方式的控制。但由于绝大部分研究者是基于库仑模型进行控制或参数辨识的，不符合实际存在的非线性摩擦模型，因而分析工具和辨识工具所预测的摩擦力参数以及有关摩擦力行为都是存在误差的。

随着电力电子学以及计算机的迅猛发展，先进的智能控制策略开始进入应用阶段，这为解决非线性摩擦力影响的问题开辟了新的途径。智能控制，尤其那些不依赖于被控对象数学模型的先进控制，对从根本上消除难以精确建模的以及难以用量化控制方法解决的非线性摩擦力的影响提供了极大的可能性。采用先进方法对非线性摩擦力进行辨识与在线补偿是一个具有挑战性的课题。富有更大挑战性的是：采用先进的不依赖于系统模型的智能控制策略来解决具有摩擦机器的控制问题。只有这样，才有可能解决摩擦力的预测模型也没有真正解决的问题。

9.2 摩擦学及实验上提出的非线性摩擦力模型

考虑图 9.1 所示的一个与弹簧常数为 k 相连、质量为 m 的物体在导轨上滑动的系统。由于运动物体与导轨间的摩擦力是随速度方向而改变的，当速度 $\dot{y}>0$ 时，库仑摩擦力为常数 $+f_c$，而当速度 $\dot{y}<0$ 时，库仑摩擦力为常数 $-f_c$，如图 9.2 所示。根据牛顿定理，此系统的运动微分方程为：

$$f(t)=m\ddot{y}(t)+f(\dot{y})+ky \tag{9.1}$$

式中，$f(\dot{y})$ 是 $\dot{y}$ 的非线性函数。

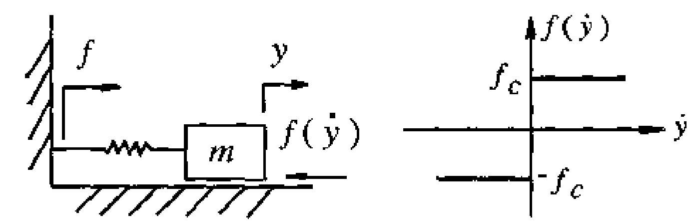


图 9.1 物体在导轨上滑动的示意图　　图 9.2 库仑摩擦力

实际上，当运动速度很慢时，摩擦力呈现出复杂的非线性变化，如图 9.3 所示。如果物体运动速度 $\dot{y}$ 很小，或 $|\dot{y}|<v_c$，摩擦力呈下降特性，即摩擦力具有负阻尼系数特性。当系统工作在图 9.3 的阴影部分时，系统会发生"爬行"现象。

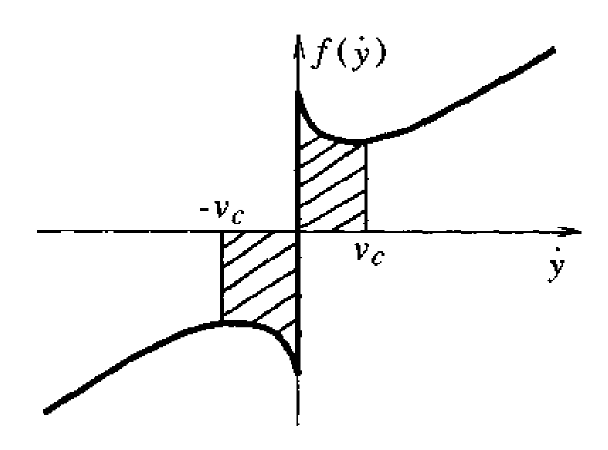


图 9.3 摩擦力特性曲线

已有不少基于实验的非线性摩擦力的模型被研究者提出。其中一个有较多共同之处的实验模型为：

$$f(\dot{y})=F_c+(F_s-F_c)\mathrm{e}^{-\phi(t,\dot{y})}+F_v\dot{y} \tag{9.2}$$

此处 F_S 为最大静摩擦力，F_c 为库仑摩擦力的最小值，不同模型的差异表现在函数 $\phi(\cdot)$ 中，有的测得 $\phi(\cdot)$ 为速度的函数，有的 $\phi(\cdot)$ 为时间的函数。总之，由上式可以看出，非线性摩擦力与速度呈现近似的指数函数关系。若把与速度成正比关系的粘滞摩擦力 $F_v\dot{y}$ 与线性系统的传递函数化为一体，则由前两项所构成的非线性摩擦力也是实际应用中常遇到的形式。

9.3 机械控制工程上采用的分析与补偿法

机械控制工程上普遍认为：最有效的消除定位或慢速控制系统中的爬行方法是切断被控过程中自身产生的自激振荡的内反馈回路。一般机械刚体（如机床工作台或其他运动部件）在润滑不足的导轨上运动时，在一定条件下会产生爬行或自激振动，另外金属切削过程或液压系统中滑阀与管道系统都有可能产生自激振动问题，这些可以应用控制理论中有关稳定性理论进行分析，设法切断内在反馈或采取其他措施，在一定程度上抑制或控制这种振动。人们将经典控制理论以及现代控制理论与机械工程科学技术有机地结合在一起，对非线性摩擦进行分析的常用方法有描述函数法及相平面法。

1) 描述函数法

描述函数法是将线性系统理论中非常有效的传递函数概念及分析方法加以推广，应用来分析非线性系统的一种方法，它的定义为：输出的一次谐波分量与输入正弦信号的复数比。描述函数法的使用是假定非线性系统满足下列条件：

①非线性系统是非时变的；

②任一非线性系统均可看成是由线性部分与非线性部分所组成，如果线性部分具有低通滤波的特性，使得高于一次的谐波均不能通过；

③输出 y 中无直流恒定分量。这样，利用描述函数来描述非线性环节，实际上是使系统线性化了。

因此，可将分析线性系统所用的奈奎斯特稳定判据、设计以及校正方法等用来对这种系统进行分析，只需要做适当的修改即可。由此可见，描述函数法是一种有效的线性化方法，这种方法较直观，但是它是一种仅适用于一定条件下的近似的方法。

利用描述函数法可以从理论上对运动机械产生爬行和自激振动的原因进行分析，并可以得出结论：产生爬行和自激振动现象的主要原因是由于摩擦力在低速时的负阻尼特性，它不能以提高自振频率来进行消除。其常用的消除办法有：

①设法消除负阻尼区，可在运动机构中增加一个正阻尼，这样从根本上消除产生爬行、自激振动的原因；

②改变润滑状态，使摩擦系数特性改变，即采用防爬行润滑油，也可以改善爬行情况；

③如果系统已存在爬行，也可外加一个高频率小振幅的强迫振动，可克服这种爬行运动，此时要求外加强迫振动的频率高于系统的自振频率。采用此方法时要注意避免这种强迫振动产生的影响。

2) 相平面分析法

所谓相平面分析法，就是用几何作图法来表示系统在各个时刻所处的状态，又可称为状态平面法。相平面法是求解微分方程的一种图解方法，以两种状态（或两项）作为坐标构成一个相平面。在相平面上作出一族相（或状态）轨迹，构成相平面图。相平面图上的每一条相轨迹表示在一定起始条件下的系统运动过程，以及状态（或项）的变化过程，从而可以得到系统的动态特性。

利用相平面法能定性地说明爬行现象发生的条件，为选择合适的参数提供定量分析的依据。用相平面法分析二阶系统是非常有力的。对于高于三阶的系统，要作出多维相空间

的图是不可能的，即使对于三阶系统要作出三维相空间图形，也是极为困难，所以一般仅用于分析二阶系统。

一般形式的二阶非线性微分方程为：

$$\ddot{x}+f(x,\dot{x})\dot{x}+\phi(x,\dot{x})x=0 \tag{9.3}$$

令

$$x_1=x$$
$$x_2=\dot{x}_1=\dot{x}$$

即

$$\frac{\mathrm{d}x_1}{\mathrm{d}t}=x_2$$

$$\frac{\mathrm{d}x_2}{\mathrm{d}t}=\ddot{x}=-f(x,\dot{x})\dot{x}-\phi(x,\dot{x})x=-f(x,\dot{x})x_2+\phi(x,\dot{x})x_1$$

所以有：

$$\frac{\mathrm{d}x_2}{\mathrm{d}t}=-\frac{f(x_1,x_2)x_2+\phi(x_1,x_2)x_1}{x_2}$$

写成如下更普遍的形式：

$$\frac{\mathrm{d}x_2}{\mathrm{d}t}=P(x_1,x_2)\text{，}\quad\frac{\mathrm{d}x_1}{\mathrm{d}t}=Q(x_1,x_2)$$

即

$$\frac{\mathrm{d}x_2}{\mathrm{d}x_1}=\frac{P(x_1,x_2)}{Q(x_1,x_2)}$$

解出上式，便可得到相轨迹的解析表达式：

$$x_2=\varphi(x_1) \tag{9.4}$$

将此相轨迹画在相平面中，便构成相平面图。

对于图 9.1 中所示具有库仑摩擦力的机械系统，此系统的运动方程式为：

$$m\ddot{y}+ky+F_c\mathrm{sgn}(\dot{y})=0 \tag{9.5}$$

取 $x_1=y$，$x_2=\dot{y}$ 为状态变量，则它的状态方程为：

$$\begin{cases}\dot{x}_1=x_2=\dot{y}\\ \dot{x}_2=-\dfrac{kx_1+F_c\mathrm{sgn}x_2}{m}\end{cases} \tag{9.6}$$

根据摩擦力的非线性特性，可将相平面分成两个区域，一是 $x_2=\dot{y}\geq 0$；一是 $x_2=\dot{y}<0$。当 $\dot{x}_2\geq 0$ 时，由（9.6）式得：

$$\frac{\mathrm{d}x_1}{\mathrm{d}x_2}=-\frac{kx_1+F_c}{mx_2} \tag{9.7}$$

解出上式并注意到 $x_1=y$，$x_2=\dot{y}$，可得到相轨迹方程为：

$$(\sqrt{\frac{m}{k}}\dot{y})^2+(y+\frac{F_c}{k})^2=R \tag{9.8}$$

式中，R 为常数，由初始状态决定。

如以 $\sqrt{\frac{m}{k}}\dot{y}$ 及 y 为坐标轴，（9.8）式是以（$-\frac{F_c}{k}$，0）为圆心的同心圆族，这便是上半平面（$\dot{y}\geq 0$）的相平面图。

同理，在 $\dot{y}<0$ 时，即下半平面的相轨迹方程为：

$$(\sqrt{\frac{m}{k}}\dot{y})^2+(y-\frac{F_c}{k})^2=R \tag{9.9}$$

图 9. 4 是根据（9. 8）式及（9. 9）式所作的相平面图。

如图 9. 4 中的起始条件已知，当起始点为 A 点时，系统的状态将沿着相轨迹 AGI（半圆）变化。然后再沿着上半平面的相轨迹 IJK（半圆）变化。当到达 $x_2=\dot{y}=0$ 以及 $|x_1|<\frac{F_c}{k}$ 时，系统在 K 点停止。

由此可见，非线性系统的静止位置是一个区域，并与起始值有关。由图 9. 4 可看出，系统是稳定的，它的最大稳态误差为 $\pm\frac{F_c}{k}$。

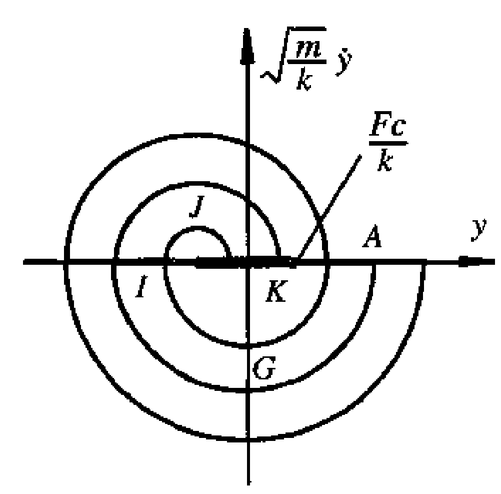


图 9. 4 系统状态的相平面轨迹图

9.4 运动控制

运动控制(Motion Control)主要包括以电机作为动力源的电气运动控制，以气体和流体作为动力源的电气控制，以燃料作为动力源的热机运动控制等等。随着微电子技术和电力电子技术以及计算机先进控制技术的发展，电气运动控制技术显示出无可比拟的优越性，从而导致运动控制的主流是电气运动控制技术，出现了电液、电气电热、机械运动控制技术。可以说运动控制是在电驱动技术研究的基础上，随着科学技术的发展而形成的一门综合性、多学科的交叉技术。几十年来，人们不断地从电力电子技术、电机技术、机械减摩技术以及先进控制策略等方面不断地寻求新的技术，以寻求达到减少执行机构的体积，减少能量损耗，改善被控系统的稳态精度和动态特性，提高控制性能的目的。尤其是 20 世纪 90 年代以来，不少研究成果已刊登在有关自动控制、机器人以及功率电子学等会

议与杂志上，由 IEEE、IFAC 分别在日本、意大利、德国和法国等国召开的专题为“先进运动控制”、“智能自动化运动控制”和“运动控制”的国际会议每年至少一次，使得运动控制在国际上逐步成为一个新的研究热点。

运动控制高性能的获得，主要是通过对执行机构、测量装置和控制策略特性的选择以及对控制策略的设计来达到的。由于摩擦力具有多变量、非线性的依赖关系，不同工况下有不同表现形式，相关的多种机理的共同作用以及运动物体自身固有的特性等原因，要得到普适的摩擦力精确模型极为困难。智能仪器和新的执行机构多是为了简化机械运动控制的实施装置，只有新的先进控制策略的设计和应用，是以消除非线性影响和提高控制性能为主要目的的。从目前的研究情况看，运动控制中先进控制策略的研究主要集中在非模型摩擦力补偿的方式上，采用如模糊控制、变结构控制、神经网络控制、重复控制以及模糊和神经网络相结合等先进控制策略。随着快速电力电子学器件、新型执行机构的不断涌现和采用，各种控制方法的相互结合以及更加先进控制策略的应用，必将使运动控制朝着快速性、精确性和鲁棒性的高性能方向发展。

9.5 运动控制中库仑摩擦力的结构分析

在运动控制的实际应用中，经常遇到诸如机器人、机床、无线电天线和天文望远镜等运动系统的速度或位置的控制。在其工作过程中存在着静、动摩擦力、粘摩擦力以及执行机构饱和等非线性特性的影响。当系统的工作点落在被控过程输入/输出特性的线性范围内时，可根据线性控制理论设计出适当的比例+微分+积分(PID)型的控制器，以满足期望的性能指标。然而实际上，我们常常会发现，金属切削机床微量进给机构在进给速度很慢时会发生爬行和自激振动；电机在低速度或变向运动时，其输入/输出特性曲线上出现死区等，这些对于有高跟踪精度和运动平稳性要求的运动控制系统来说，带来了控制上一定的困难，因为采用单纯的 PID 型控制器，其参数在稳态误差与运动平稳性的兼顾上是有矛盾的。机械控制工程上普遍认为最有效地消除定位或变速控制系统中的爬行与自激振荡的方法是切断被控过程中自身产生的自激振荡的内反馈回路，主要考虑通过低摩擦机器的设计、润滑的选择以及利用描述函数或相平面法对系统稳定性的分析而选择使系统稳定的机械系统的参数。控制工程的设计则主要是基于摩擦力模型的补偿技术。在系统稳定的前提下，从误差和平稳性的综合角度去考虑问题，采用自适应控制策略去补偿时变的非线性摩擦力。

本节对机电执行机构中存在的各种摩擦力进行分析，分析库仑摩擦力在二阶系统中对不同形式的线性补偿器所产生的稳态误差，并指出其减少和消除的途径。指出直接的力矩前向补偿法是最有效的消除摩擦力的方法，提出一种非线性摩擦力模型与其参数确定的实验室方法，最后给出采用自适应参数辨识法直接对库仑常数摩擦力值的辨识公式。

在机电工程中，以直流电机为例，其机电动态特性的运动可用下列微分方程描述：

$$T = J\ddot{\theta} + B\dot{\theta} + T_f \tag{9.10}$$

其中 θ 为角位移输出，J 为总惯量，B 为粘摩擦系数，T 为控制输入力矩，T_f 为静动摩擦力矩。

众所周知，机电系统中的摩擦力是由静摩擦力、库仑摩擦力和粘摩擦力组合而成的(见图 9.5)，因为粘摩擦力是与系统的速度成正比的，是速度的线性函数，人们常常把它与线性系统的建模化为一体，成为线性系统的一部分。而分离出粘摩擦力后的静动摩擦力 T_f，是我们常常研究的对象。静摩擦力仅存在于 $\dot{\theta}=0$ 时刻，且它的幅值总是大于库仑摩擦力。一旦 $\dot{\theta}\neq 0$，就只有库仑摩擦力作用于执行机构，其值为一常数 T_c，作用力的方向总是与运动方向相反。当工作点落入线性区时，有下式成立：

$$T_f = T_c\,\mathrm{sgn}(\dot{\theta}) \tag{9.11}$$

其中，sgn(·)为符号函数。若设控制器的传递函数为 $G_c(s)$，则具有干扰 T_f 的负反馈控制系统的结构图如图 9.6 所示。

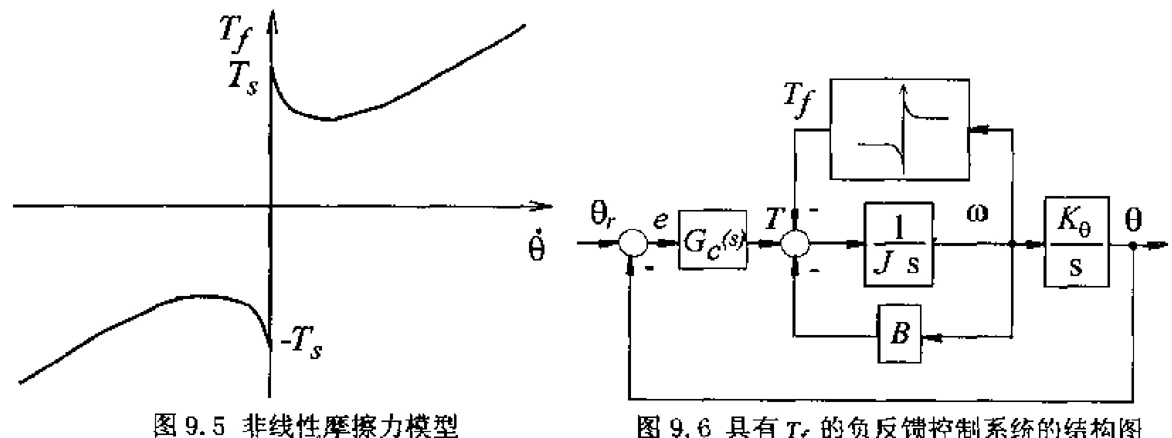


图 9.5 非线性摩擦力模型

图 9.6 具有 T_f 的负反馈控制系统的结构图

对于 PID 型补偿器结构，我们通过令参考输入 $\theta_r=0$，分析仅由 T_f 单独作用时，在不同补偿器结构的作用下，系统的稳态误差 e_∞ 的值来看其补偿效果。为了分析简便，对 T_f 仅考虑库仑摩擦力 T_c 的影响。当补偿器仅为比例项，即 $G_c(s)=K_p$ 时，输入 $T_f(s)$ 与输出 $\Theta(s)$ 之间的传递函数为：

$$\frac{\Theta(s)}{T_f(s)}=\frac{1}{Js^2+Bs+K_p}$$

此时，$T_f(s)$ 与误差 $E(s)$ 之间的传递函数可以写为：

$$\frac{E(s)}{T_f(s)}=-\frac{\Theta(s)}{T_f(s)}=\frac{1}{Js^2+Bs+K_p}$$

由终值定理可以求得由(9.11)式确定的库仑摩擦力所产生的稳态误差为：

$$\begin{aligned} e_\infty &= \lim_{s\to 0} sE(s) \\ &= \lim_{s\to 0}\frac{-s}{Js^2+Bs+K_p}\cdot\frac{\pm T_c}{s} \\ &= \mp\frac{T_c}{K_p}=\begin{cases} -\dfrac{T_c}{K_p} & 当\dot{\theta}>0 \\ +\dfrac{T_c}{K_p} & 当\dot{\theta}<0 \end{cases} \end{aligned} \tag{9.12}$$

此时补偿器的控制输出为：$T_{\infty}=K_p \cdot e_{\infty}=\mp T_c$。

由此可以看出，稳态时，比例补偿器提供的控制力矩为$\mp T_c$，其幅值与干扰摩擦力矩相同，方向相反，以此来将其抵消。系统在 T_f 的作用下的稳态输出为：$\theta_{\infty}=-e_{\infty}=\pm T_c / K_p$，即比例补偿器$G_c(s)=K_p$，将干扰值减少了 $1/K_p$。这也正是负反馈作用的结果：降低了干扰的影响，但没有完全将其消除。由（9.12）式可以看出，可以通过加大比例常数K_p的值来减少系统的稳态误差，然而，增加K_p的同时，也增大了系统响应的波动性。

虽然在比例补偿器的作用下，只要$K_p>0$，闭环系统为二阶稳态系统，但随着K_p的增加，所产生系统响应的误差与波动并不是我们所期望的，所以仅用比例补偿器来消除位置控制系统中的库仑摩擦力的影响是不够理想的。可以用以改进的办法之一是采用比例加积分的 PI 补偿器。此时，比例控制作用趋向于稳定系统，而积分控制作用趋向于减少对不同输入响应的稳态误差。只要系统中存在误差信号，补偿器就产生一个将其减少的力矩，且同时使系统稳定。

采用与上面同样的分析方式对$G_s(s)=K_p(1+1/T_i s)$时控制系统的稳态误差进行分析。当$\theta_r=0$时，只要控制系统特征方程$Js^3+Bs^2+K_p s+K_p/T_i=0$的根具有负实部，由终值定理可以求得对于幅值为$|T_c|$的库仑摩擦力矩干扰的稳定系统的稳态误差为：

$$
\begin{aligned}
e_{\infty} &= \lim_{s\to 0} sE(s) \\
&= \lim_{s\to 0} \frac{-s^2}{Js^3+Bs^2++K_p s+\dfrac{K_p}{T_i}} \cdot \frac{|T_c|}{s} \\
&= 0
\end{aligned}
\tag{9.13}
$$

由此可见，采用 PI 补偿器，可以将控制系统中的库仑摩擦力完全消除。但是值得注意的是，由于在比例器中加入了积分器，使原先的二阶系统变为三阶系统，所以当K_p取值较大时，有可能使闭环系统的特征多项式的根变为正实部而造成系统的不稳定。所以对K_p的选值要加以限制。另外，如果只采用单独的积分器$G_s(s)=K_p/s$作为补偿器，系统将变为不稳定系统。因为此时闭环系统的特征多项式变成了$Js^3+K_p s^2+K_p=0$，总存在正实部的根。这样的不稳定的系统在实际中是不能使用的。

如果在比例器的基础上加微分器，可以使得补偿器具有更高的灵敏度。微分器一方面

对误差的变化率给予响应，且能够在误差的幅值变大之前产生一个校正量，从而预计误差值，提前给予一个校正值，并能够增加系统的稳定性。另一方面，虽然微分控制不直接对稳态误差有影响，它可以增加系统的阻尼，从而允许使用较大的放大增益 K_p 而间接地改善系统的稳态精度。所以对于位置控制系统来说，如果不要求稳态误差一定为零，则可以使用 PD 补偿器，这样可以使用较大的 K_p，同时因被控系统为正系数的二阶系统而总是稳定的。

9.6 基于模型的摩擦力前向补偿器的设计

上节分析了不同类型的补偿器对常数库仑摩擦力的补偿作用，这是通过闭环负反馈补偿器对干扰进行补偿与消除的技术。实际上，对库仑摩擦力的最有效的补偿方法应当是直接力矩的前向补偿法。只要能够预测出干扰的数值，即可在前向回路中的正常控制量上加上一个与干扰同值但符号相反的作用将其抵消，从而使干扰的作用在被传感器感应并放大之前就将其消除。

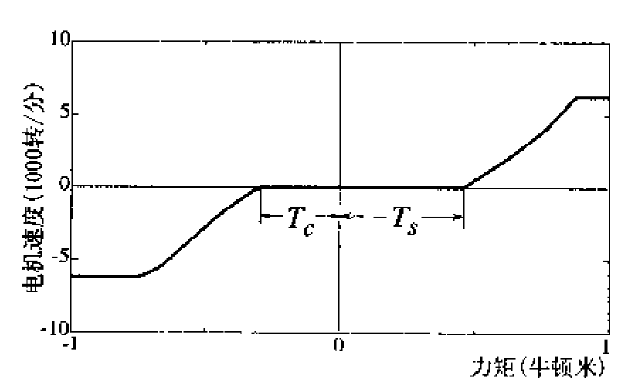


图 9.7 直流电机静态力矩—速度特性曲线

运用前向补偿的关键在于对摩擦力矩的合理估值。对于仅采用库仑摩擦力作为模型的估值，可以通过下述实验进行。向被控过程输入初始值为零、斜率为正的斜坡信号，并测量过程速度输出的响应。由此得到其“力矩—速度”的输入/输出的静态特性曲线如图 9.7 所示，T_c 的值是通过当速度由反向正斜率斜坡信号上升，变为 $-\dot{\theta}=0$ 时力矩的输入值来确定的。此值所对应的输入力矩即为库仑摩擦力矩的估计值 $\hat{T}_C$。

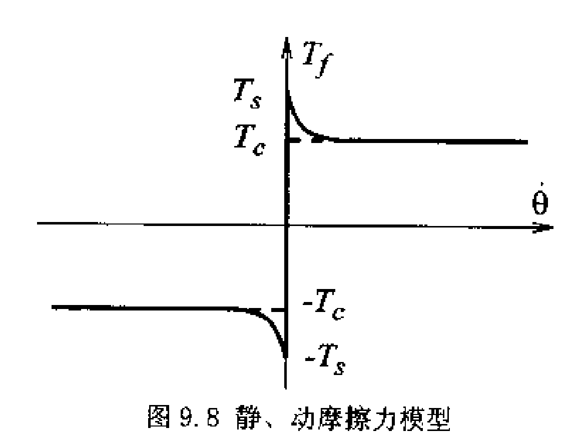


图 9.8 静、动摩擦力模型

实际上，采用不同斜率的斜坡输入所得到的 $\hat{T}_C$ 不尽相同，这可以由图 9.8 来解释：静摩擦力矩虽然仅存在于 $\dot{\theta}=0$ 时刻，在 $\dot{\theta}$ 由零变为非零的暂短的时间中，静摩擦力矩不可

能立刻降为零，而是有一个降至零的过渡过程。这个过渡过程的快慢程度与速度成近似的指数关系。所以输入对应于不同的斜率的斜坡，得到克服静摩擦力而使执行机构开始运动的时间长短不同。斜率大的较早地达到稳态库仑摩擦力常值。所以此法只能测出 T_c 的一个估计值$\hat{T}_c$，估值误差只能通过闭环控制器进一步加以消除。

由图 9.8 可以看出，更合理的摩擦力模型应同时考虑静、动摩擦力影响的非线性摩擦力模型。一个与速度成指数关系的摩擦力模型可以为：

$$T_f = T_c + (T_s - T_c)\mathrm{e}^{-\alpha\dot{\theta}} \tag{9.14}$$

模型中的T_S为静摩擦力矩，α为时间常数，且根据具体被控过程由实验确定。T_S是利用图 9.7 中静态力矩—速度特性曲线，使输出从零转为大于零时的输入值来确定的。值得注意的是，利用此非线性模型进行补偿时，它仅需应用于$|\ddot{\theta}| > 0$段，即正、反向加速段，在$|\ddot{\theta}| < 0$段的摩擦力补偿仍是采用(9.11)式。采用 (9.14)式模型补偿的效果显然优于(9.11)式的效果。进行非线性摩擦力的补偿以及系统的控制规律可以通过比如数字信号处理器的控制系统来实现。

9.7 线性模型的参数辨识

众所周知，经典控制理论的实际应用是基于被控对象的数学模型。然而在许多情况下，被控对象的数学模型很难精确获得，或者对象模型的参数在工作过程中随时间发生变化，另外出于系统仿真的需要，使系统模型的建立和系统参数的辨识常常成为能否成功地应用控制理论的关键之一。它的实际应用已遍及各个领域。本节首先给出对人们在实际中常用的各种不同的模型的结构及其功能，并与控制界常用的 MATLAB 环境下系统辨识工具箱中所采用的模型相比较，指出各自的特点，最后给出常数库仑摩擦力的辨识方法。

所谓的线性模型的“辨识”就是在所测取的输入和输出数据的基础上，从一组给定的模型中，确定一个与所测系统等价的模型。辨识过程把待辨识过程看作是一个“黑箱”，它只考虑过程的输入/输出特性，而不强调过程的内部机理。系统辨识的方法有许多种，可以有各种不同的分类法。若根据所涉及的模型形式，辨识的方法可分为非参数模型辨识和参数模型辨识两种；若根据计算机与系统之间的不同连接方式，辨识的方法又可分为在线辨识和离线辨识；如果根据不同的辨识基本原理，常用的参数模型辨识方法又可分为三种：最小二乘法、梯度校正法和极大似然法。

9.7.1 基于不同原理的辨识方法

对于待辨识过程，其输入$u(k)$和输出$y(k)$是可以观测的；通常描述过程输入/输出特

性的模型$G(z^{-1})$可以表示成$G(z^{-1})=\dfrac{B(z^{-1})}{A(z^{-1})}$，其中

$$\begin{cases}A(z^{-1})=1+a_1z^{-1}+a_2z^{-2}+\cdots+a_{n_a}z^{-n_a}\\B(z^{-1})=b_1z^{-1}+b_2z^{-2}+\cdots+b_{n_b}z^{-n_b}\end{cases}$$

过程的输出除了受输入$u(k)$作用之外，往往还受到其他一些不确定因素的影响，通常把这些不确定的影响都归结成附加噪声$n(k)$。当$\{n(k)\}$是平稳的随机序列，且均值为零、谱密度是$\cos\omega$的有理函数时，$n(k)$可以表示成$n(k)=N(z^{-1})v(k)$，其中，$v(k)$为白噪声；$N(z^{-1})$是噪声模型，通常可表示成$N(z^{-1})=\dfrac{C(z^{-1})}{D(z^{-1})}$，式中：

$$\begin{cases}C(z^{-1})=1+c_1z^{-1}+c_2z^{-2}+\cdots+c_{n_c}z^{-n_c}\\D(z^{-1})=1+d_1z^{-1}+d_2z^{-2}+\cdots+d_{n_d}z^{-n_d}\end{cases}$$

在传统的辨识方法中，常用的基于差分方程的参数模型辨识有以下几种。

1）最小二乘类算法

最小二乘类参数辨识方法，包括最小二乘法、增广最小二乘法、广义最小二乘法、辅助变量法等，其辨识方法就其原理来说是相同的，只是各种方法所用的模型结构不一样。例如：

最小二乘法所用模型是：

$$A(z^{-1})y(k)=B(z^{-1})u(k)+v(k) \tag{9.15}$$

增广最小二乘法所用模型是：

$$A(z^{-1})y(k)=B(z^{-1})u(k)+C(z^{-1})v(k) \tag{9.16}$$

广义最小二乘法所用模型是：

$$A(z^{-1})y(k)=B(z^{-1})u(k)+\frac{1}{D(z^{-1})}v(k) \tag{9.17}$$

等等。可见不论采用什么样的辨识方法，所用的过程模型总是一样的，只是噪声模型有所不同。

最小二乘法是最基本的，也是应用最广泛的一种方法。上面所提的其他方法都是以最小二乘原理为基础的。最小二乘法大约是 1795 年高斯在他的著名的星体预报研究中提出的。由于最小二乘法严密简单，编程容易，成为估计理论的奠基石，得到广泛的应用。最小二乘法的基本结果有两种形式，一种是经典的一次完成算法；另一种是递推算法。后者常使用于计算机在线辨识。

2）最小二乘法的解

设过程的输入/输出关系可以描述成如下的最小二乘形式：

$$y(k)=\phi^{\tau}(k)\theta+v(k) \tag{9.18}$$

其中，$v(k)$为白噪声，$\phi(k)$为可观测的数据序列，θ为被估计参数序列，分别表示为：

$$\begin{cases}\phi(k)=[-y(k-1),\cdots,-y(k-n_a),u(k-1),\cdots,-u(k-n_b)]^{\tau}\\ \theta=[a_1,a_2,\cdots,a_{n_a},b_1,b_2,\cdots,b_{nb}]^{\tau}\end{cases}$$

最小二乘算法的原理就是根据数据序列$\{y(k)\}$和$\{\phi(k)\}$，极小化下列准则函数：

$$J(\theta)=\sum_{k=1}^{L}[y(k)-\phi^{\tau}(k)\theta]^2 \tag{9.19}$$

使$J(\theta)$为极小时的θ估计值记作$\hat{\theta}$，并被称为参数θ的最小二乘估计。

上述过程表明，未知模型参数θ最可能的值是在实际观测值与计算值之间累次误差的平方和达到最小值处。

标准最小二乘问题的解为：

$$\hat{\theta}=(\Phi^{\tau}\Phi)^{-1}\Phi y \tag{9.20}$$

最小二乘递推算法为：

$$\begin{cases}\hat{\theta}(k)=\hat{\theta}(k-1)+K(k)[y(k)-\phi^{\tau}(k)\hat{\theta}(k-1)]\\ K(k)=P(k-1)\theta(k)[\theta^{\tau}(k)P(k-1)\theta(k)+1]^{-1}\\ P(k)=[I-K(k)\theta^{\tau}(k)]P(k-1)\end{cases}$$

3）增广最小二乘法的解

增广最小二乘法是最小二乘法的一种简单推广，它把噪声模型的辨识在模型中同时进行考虑。增广最小二乘法的解法与最小二乘法的解法基本相同，过程的输入/输出关系同样可以描述为（9.16）式所表现的最小二乘形式。其中：

$$\begin{cases}\phi(k)=[-y(k-1),\cdots,-y(k-n_a),u(k-1),\cdots,\\ \qquad -u(k-n_b),v(k-1),\cdots,-v(k-n_c)]^{\tau}\\ \theta=[a_1,a_2,\cdots,a_{n_a},b_1,b_2,\cdots,b_{nb},c_1,c_2,\cdots,c_{n_c}]^{\tau}\end{cases}$$

$\phi(k)$中包含不可测量噪声$v(k-1),\cdots,-v(k-n_c)$，可按下式求得相应的估计值代替：

$$\begin{cases}\hat{v}(k)=0 & k\le 0\\ \hat{v}(k)=y(k)-\phi^{\tau}(k)\hat{\theta}(k-1) & k>0\end{cases}$$

4）广义最小二乘法

广义最小二乘法的基本思想是对输入/输出数据先进行一次滤波处理，然后利用普通最小二乘法对滤波后的数据进行辨识，模型如（9.17）式所示。由于其中的噪声模型$D(z^{-1})$未知，所以需要首先利用最小二乘法获得$D(z^{-1})$中参数的估计。然后令：

$$\begin{cases}y_f(k)=D(z^{-1})y(k)\\ u_f(k)=D(z^{-1})u(k)\end{cases}$$

则(9.17)式转化为标准的最小二乘形式（9.15）式。

（1）辅助变量法

辅助变量法的基本思想是适当地选择辅助变量，使之满足一定的条件，以获得参数的

无偏一致估计。

$$A(z^{-1})y(k)=B(z^{-1})u(k)+e(k) \tag{9.21}$$

其中，$e(k)$ 为有色噪声。

由于 $e(k)$ 为有色噪声，直接利用最小二乘法不能获得参数的无偏一致估计，所以需要先利用辅助变量将（9.21）式转换成前面已经介绍过的易求解的形式，然后利用最小二乘法进行参数估计。

（2）梯度下降法

梯度下降法利用最速下降法的原理，沿着误差准则函数关于模型参数的负梯度方向，逐步修改模型的参数估计值，直至误差准则函数达到最小值。

$$y(t)=\phi^{\tau}(k)\theta \tag{9.22}$$

迭代公式为：

$$\hat{\theta}(k+1)=\hat{\theta}(k)+R(k)\phi(k)[y(k)-\phi^{\tau}(k)\hat{\theta}(k)]$$

其中，$R(k)$ 为权矩阵，其作用是用来控制各输入分量对各权估计值的影响程度。

采用梯度下降法进行参数估计的原理完全不同于最小二乘法，它的特点是计算简单，可以用于在线的实时辨识。

（3）极大似然法

极大似然法是根据极大似然原理，通过极大化似然函数来确定模型的参数。

$$A(z^{-1})y(k)=B(z^{-1})u(k)+C(z^{-1})v(k) \tag{9.23}$$

极大似然法的基本思想是构造一个以数据和未知参数为自变量的似然函数，并通过极大化这个似然函数，获得参数的估计值。极大似然法通常要求具有能够写出输出量的条件概率密度函数的先验知识，因而，极大似然法的计算量比较大。而前面介绍的两种方法：最小二乘法与梯度下降法不仅计算简单，而且具有良好的无偏性和一致性，对噪声特性的先验知识要求也不高。

9.7.2 MATLAB 中系统模型辨识的描述方法

与根据算法原理分类不同，控制界中常用的MATLAB环境下系统辨识工具箱中的模型是根据不同的模型结构来进行分类的。

1）自回归模型结构（ARX Structure）

$$A(q)y(t)=B(q)u(t-nk)+e(t) \tag{9.24}$$

其中：

$$\begin{cases}A(q)=1+a_1q^{-1}+a_2q^{-2}+\cdots+a_{na}q^{-na}\\B(q)=b_1+b_2q^{-1}+b_3q^{-2}+\cdots+b_{nb}q^{-nb+1}\end{cases}$$

注意，在MATLAB中，符号$e(t)$被用来表示白噪声。显然，这一模型与标准的最小二乘模型（9.15）式完全一致。其求解方法在MATLAB中，求解线性方程组的算符“\”可以直接给出参数的最小二乘估计。

2）自回归滑动平均模型结构（ARMAX Structure）

$$A(q)y(t)=B(q)u(t-nk)+C(q)e(t) \tag{9.25}$$

其中，$C(q)=1+c_1q^{-1}+c_2q^{-2}+\cdots+c_{nc}q^{-nc}$。这一模型与增广最小二乘模型（9.14）式完全一致。

在MATLAB系统辨识工具箱中，是通过采用Gauss-Newton 搜索算法最小化误差准则函数来获得参数估计的。其做法是在Gauss-Newton方向上用二分法搜索１０次，直到找到一个更小的误差；否则，转到梯度方向搜索来获得更小误差。

3）输出误差结构（Output-Error Structure）

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+e(t) \tag{9.26}$$

其中，$F(q)=1+f_1q^{-1}+f_2q^{-2}+\cdots+f_{nf}q^{-nf}$

此算法基本上和ARMAX Structure相同，也只是误差准则与梯度的计算略有不同。

4）Box-Jenkins 模型结构（*Box-Jenkins* Structure）

$$y(t)=\frac{B(q)}{F(q)}u(t-nk)+\frac{C(q)}{D(q)}e(t) \tag{9.27}$$

其中：$D(q)=1+d_1q^{-1}+d_2q^{-2}+\cdots+d_{nd}q^{-nd}$。

不难看出，这是一种最一般的结构，前三种结构都是它的特例。此模型算法基本上和*ARMAX* Structure相同，只是误差准则与梯度的计算上有不同。

另外，还有所谓的“辅助变量估计”（auxvar.m）和“预测估计”（pem.m）。对于实际系统的辨识来说，应该选用什么样的模型并没有一个可遵循的原则。一般来说，可先采用简单的模型，在获得辨识结果后，检验模型的可信度，如果不能满足要求就需要换用其他种模型，这时所用的辨识方法自然也就不同。也就是说，解决一个实际问题，到底应该采用那种辨识方法，主要取决于模型类的选择，而模型类的确定往往需要通过多次的实验比较后才能确认。

9.7.3 库仑摩擦力参数的辨识

库仑摩擦力参数 T_c 的辨识既可以通过实验求出，也可以把 T_c 作为待辨识的参数之一与其他系统待辩参数 A、B 同时进行辨识，若采用最简单的最小二乘模型，此时的系统模型可以写成：

$$A(q^{-1})y(k)=B(q^{-1})u(k)+T_c \tag{9.28}$$

则有：

$$y(k)=\Phi^{\mathrm{T}}\theta+e(k)$$

其中：

$$\Phi=\left[y(k-1)y(k-2)\ldots y(k-n_a)u(k-1)u(k-2)\ldots u(k-n_b)1\right]$$

$$\theta = \left[a_1 a_2 .. a_{n_a} b_1 b_2 .. b_{n_b} T_c\right]$$

以此方式，可以采用标准的最小二乘参数辨识方法，利用现有的参数辨识软件如MATLAB 参数辨识工具包，并在系统的工作点施加伪随机二进制信号(PRBS)进行离线参数辨识。

9.8 运动控制中的机电控制系统

随着微电子学、计算机以及数据处理技术的不断进步，控制系统与机械系统得到进一步的有机结合，使得在对机械系统的实时控制上能够很容易地运用十分复杂的控制算法，同时也使整个机电计算机控制系统更具有智能性和易使用性，这些都集中地表现在机电一体化的计算机控制系统之中。

机电计算机控制系统一般由三部分组成：

①机械系统：传感器、执行机构、元器件和组合件；

②电子学技术：微处理器技术、数字通讯、接口技术和系统结构；

③先进的计算机控制：系统理论、信息技术、发展的软件设计、仿真以及机电计算机控制系统的实施。

对机电计算机控制系统的三个组成部分，必须从设计的最初阶段就进行整体化的考虑。即要考虑机械系统、电子学技术及计算机控制的各个方面。

9.8.1 机械系统

采用机电计算机控制系统对一个复杂的机械系统进行控制，虽然只需要用到被控过程的输入、输出数据，但对机械系统中每个部件的性能及作用的正确理解与认识，对于正确的系统建模和控制器的设计是非常必要的，并可以加速设计过程。

在典型的伺服控制系统中，机械系统是由通过电子集成电路控制的执行机构组成的，并通过功率放大器将由计算机产生的控制信号传送给执行机构。目前最常使用的功率放大器是脉宽调制功率放大器(PWM)，用以减少功率损耗。在机械传感器的测量中，测速器是用来测量电机的旋转速度。常用的测速器为位移测量计，其他方法有用电磁感应法、磁平衡法和磁应力法设计的测速器或精度很高的光电编码器。在全过程控制系统中，除需使用传感器测量物理量并产生各种信号外，还需将它们转变成适当的形式，以便于进行元部件间的相互联接。这些是由信号转换器或信号调节器来完成的。最后，所有这些部件都需要电源。

9.8.2 电子学技术

在机电计算机控制系统中，微计算机有时被用来当作控制器。它的操作实现三个功能：读取输入、控制律的计算和结果的输出。控制器的控制律是通过软件程序执行的。首先，

A/D 转换器将执行机构的模拟信号转换成数字信号，并读入控制器中。在进行控制策略的计算后，信号被转化成模拟量后输出。这个操作过程可以在几微秒的间隔内完成，并不断地重复。在每一个间隔之间，输出信号是以不变的常数形式一直保持到下一个周期。目前国际上除了采用微控制器产生控制信号外，数字信号处理器以其速度快、功能强和易使用等优点而被广泛地应用。所以对数字信号处理器的深入了解是设计一个机电计算机控制系统的基本要求。这些方面包括其结构、特性、功能及接口。除了这些基本知识，为了执行好控制策略，设计者必须从技术与应用方面掌握各指令的功能及用法，各种工作方式、中断方式及应用软件库中各种通用函数的调用。

先进的电子技术使系统更加集成化，其特性之一就是通过接口能够使所有的元部件同时很好地协作。计算机与其他装置之间的接口是借助于标准的联接器、信号线、电子性能以及数据交换方式进行的。标准化器件之一是 RS-32C 接口，另一个是 IEEE-488，执行机构产生的模拟信号与由计算机产生的数字信号之间的接口是 A/D 和 D/A 转换器，数据是在装置之间通过异步或同步方式进行传送的。

9.8.3 先进的计算机控制

常规的控制设计方法是基于一个线性时不变的假定设计出的常系数算法。在工业应用中使用的大部分控制器是 PID 型，用来调节 PID 控制器有多种不同的方法，Ziegler-Nicholas 法是较流行的。对参数随时间变化的系统也可采用自适应控制算法。自适应控制系统是基于对一个传递函数的递推估计，然后进行控制器的设计。这里实际上主要循环着两个过程：被控过程模型参数的估计以及所应用的控制算法的计算。根据实际情况，可以选择各种模型并通过软件进行系统建模。众所周知的有关参数辨识的递推算法有：最小二乘法、辅助变量法、最大似然法等。对于一个时不变的系统，一般过程的参数辨识与分析以及控制算法的设计常常采用离线辨识；参数辨识是通过计算机辅助设计包（CAD）进行的。自适应控制器的设计可基于最小方差、极点配置或 LQG 设计来自动调整调节器参数。控制器也允许进行前向补偿或控制。

模型仿真通常包括一组微分方程或差分方程。传感器、机械部件的执行机构、转换器、滤波器及信号处理器可以用适当的数学模型进行仿真，然后通过计算机的仿真技术对整个系统的行为进行仿真。如今，软件可以对仿真提供各种的方法，既能够对微分方程（连续系统），又能够对差分方程（离散系统）和含有离散与连续方框图的混合系统进行仿真。由于仿真参数的正确性是获得实际精确控制结果的保证，所以全面地理解机械系统以及电子学系统每一个部件对仿真过程是相当重要的，正确地理解和使用仿真软件可以加速设计过程。

在机电计算机控制系统的设计与实施过程中，软件的质量是系统是否成功的重要因素。可以说，正是有了强有力的软件包，我们才能加快设计过程并使控制系统更加集成化。一个机电计算机控制的软件设计不是独立的，而是软硬件之间、机械电子之间的连接与并行操作的结合。软件设计的核心是一个支持实时处理与大量批处理的操作系统。一个完整的支持软件包在控制系统中是相当重要的，它包括语言处理器、交叉屏幕编辑器以及可用

于任何类型应用的标准工具包。应用软件是对控制系统应用专门开发出的一组程序，它包括数据采集、模型辨识、控制器设计、控制仿真、程序交换以及控制执行等。例如，由美国 Math Works 公司出品的 MATLAB 软件及其工具包是一个容易使用，集中了数学分析、混合运算、信号处理以及图形仿真的高性能的软件包。其环境允许用户用图形表达各种数学公式，并产生相应的用高级语言写成的子程序。已经发展的许多功能包括各种矩阵的计算、图形显示功能、控制系统工具箱、鲁棒控制工具箱、系统辨识工具箱、信号处理工具箱、状态空间控制工具箱、优化工具箱、神经网络工具箱、频率响应控制工具箱、模糊逻辑工具箱等。

数字信号处理器发展起来的应用程序可以用来产生、测试以及调试用户应用程序。例如在 DSP32C 中，自身的支持软件可在微计算机上运行，它在 MS-DOS 操作系统下工作，其软件包括汇编器、连接编辑器、仿真器以及工具包。其汇编语言允许程序用近似 C 语言方式进行书写，使用户很容易掌握。

9.8.4 自适应运动控制应用的例子

运动控制涉及线性和非线性过程，常用电机作为执行机构，并采用传感器进行状态变量的测量。运动控制的典型特性及要求是：

①被控过程通常是稳定的；

②系统具有非线性，诸如静、动摩擦力，惯性负载，执行机械饱和特性等等；

③要求具有快速的动态响应、高质量的稳态精度和抗非线性干扰的能力。

在本应用中，控制系统跟踪一个受非线性摩擦力干扰的直流电机的速度。机电计算机控制系统是基于 PC-386，一个嵌于计算机中由 AT&T 公司生产的数字信号处理器 DSP32C 板以及接口电路。图 9.9 给出了整个控制系统结构图。采用基于摩擦力矩模型的直接在线地前向补偿与常规的负反馈控制器相结合的控制策略。线性系统参数的辨识采用 ARMAX 动力学模型。采样时间为 5 毫秒。辨识后所获得的被控系统的线性模型为：

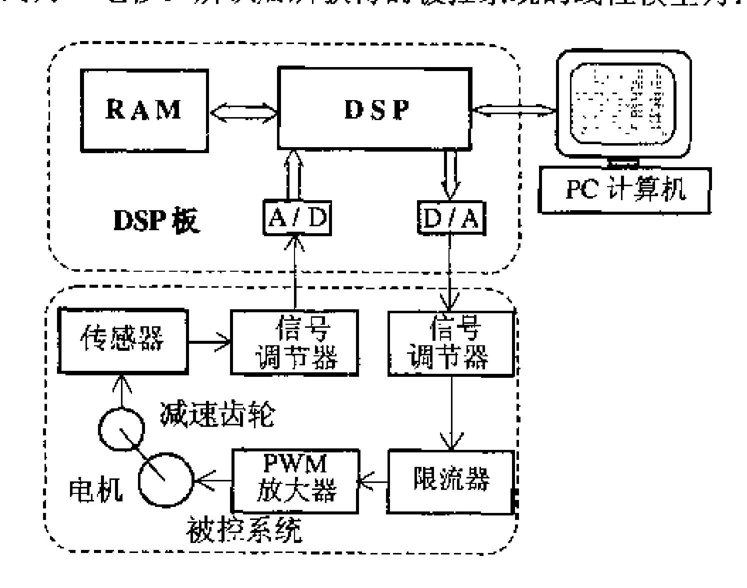


图 9.9　被控直流电机的机电计算机控制系统结构图

$$P = 0.0074z^{-1}/(1-0.9928z^{-1})$$

以此模型进行反馈控制器设计与仿真。用于前向补偿的非线性摩擦力模型，是通过对被控电机实际的力矩—速度的测试曲线，经过设计、仿真并实际验证后获得的，形式为：

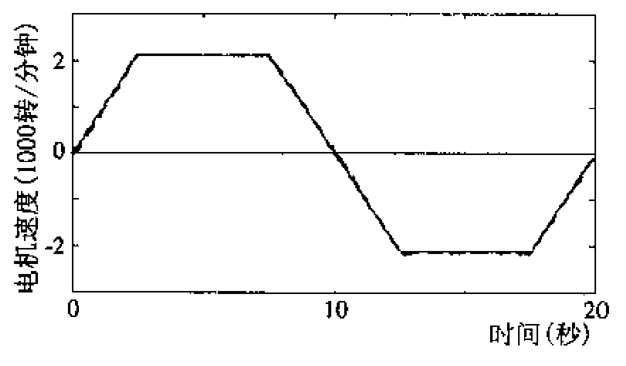
$$\hat{T}_f(\omega) = 12300\text{sgn}(\omega) + 1700\text{sgn}(\omega)\cdot e^{-0.0001129|\omega|}$$

其中，sgn 为符号函数。力矩单位为数字计算机中运算用的“数字”。

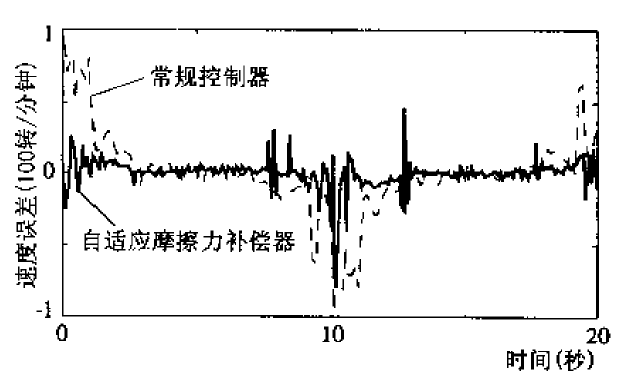
以此方式产生一个稳定的具有对摩擦力干扰进行前向在线补偿的负反馈控制。

控制信号由 DSP32C 的汇编软件执行，实时控制策略是通过 DSP32C 进行操作的，然后通过 D/A 转换，并通过功率放大后生成电机的力矩控制。速度的测量由测速计执行，并通过一个 16 位 A/D 转换器变成数字信号后送入计算机。在离线的模型辨识、仿真以及设计过程中，使用 MATLAB 函数文件对数据进行处理，并进行参数估计、数据滤波和控制算法的仿真。系统控制程序是由 C 语言写成的，用它来控制检查输入输出操作，并将在 DSP32C 中产生的数据文件格式与 MATLAB 的数据格式进行相互转换。

实际的机电计算机系统对梯形输入信号的响应曲线如图 9.10(a)所示。图中示出了带有自适应摩擦力补偿的跟踪响应以及仅采用常规控制器控制的速度跟踪响应。常规控制器是用极点配置法设计的 PI 控制器。为了对比两者的控制效果，图 9.10(b)中给出了相应的速度响应误差曲线。从中可以明显地看出，由常规控制器所无法很好跟踪的、由非线性摩擦力影响所造成的零附近的跟踪误差，通过自适应摩擦力前向在线补偿得到了很大的改善。



(a) 具有自适应摩擦力补偿与仅有常规控制器控制的速度控制实验响应曲线



(b) 具有自适应摩擦力补偿与仅有常规控制器控制的速度响应误差

图 9.10 控制系统结果对比图

由计算机或微处理机控制形成的机电计算机控制系统不是新的工程领域，它可以被看作一个由机械，仪器、先进控制工程以及微处理器组成的集成化协作工程。所有这些都基于执行机构性能的不断改善、电力电子学的进步、先进的控制策略的设计、数字信号处理器的发展以及计算机软件和商业标准软件的支持。随着科学技术各方面的不断发展，一定会有越来越多的具有集成化和易使用的机电一体化产品进入我们的日常生活中。

9.9 非线性直流电机仿真模型系统的建立

物体运动在零速度附近产生的非线性摩擦力，是运动物体所固有的特性，存在于定位控制、低速度和速度变向的伺服系统中。即使改变阻尼、刚度、边界润滑等因素，也很难消除由非线性摩擦力所引起的振动或“死区”现象。这类问题不仅在滑动或电机旋转的机械中广泛地存在，而且在机器人等高精度自动化设备中也成为亟待解决的棘手问题。在现代控制器的设计过程中，被控系统的仿真已成为不可缺少的重要一环。它对系统的分析和研究、控制器的设计与参数调试、甚至实际控制性能的好坏都起到关键性的作用。虽然不依赖于模型的新型控制策略在不断地涌现，但即使如此，在设计阶段也离不开仿真系统实验的调试与证实。在一个精确符合实际系统的仿真模型上所获得的仿真实验结果与实际控制效果相差无几。

本节从简单的直流电机入手，着重考虑其运行在零速度附近所产生的非线性摩擦力的影响，结合已获得的研究成果，利用 SIMULINK 软件，在 MATLAB 环境下具体深入地建立起用于控制器设计和系统仿真的非线性直流电机的模型系统，在验证所建非线性系统有效性的同时，也显示出 SIMULINK 软件的强大功效。

9.9.1 被控过程线性段模型的参数辨识

1) 实际被控系统的输入/输出特性

实际控制系统包括一台 Pentium 200 计算机、一块内置于计算机的 12 位 A/D, D/A 转换板、PWM 功率放大电路、直流力矩电动机及用于速度反馈的直流测速发电机。模拟电压输入范围与输出控制电压范围均为 [-5, +5]V，经 12 位 A/D, D/A 转换后的数字量范围均为 [-2048, +2048]，为了方便起见，所用输入/输出单位均采用数字量。被控系统的模型包括除计算机外的所有部件的集合。

为了观察实际系统在零速度附近的输入/输出特性，向系统输入斜率为 1 的斜坡信号，并测量系统输出，所获对逐渐上升增加及下降减少的输入信号的响应输出如图 9.11 所示。从图中可以清楚地看到，系统在输入信号小于某个值的范围内无输出信号，即出现所谓的“死区”现象；而在输入信号的绝对值大于一定数值后，其输出也不再随输入信号的变化而变化，出现了所谓的“饱和”现象。这两者都是机电控制系统中的典型非线性现象。另一个值得注意的问题是：对于上升和下降的输入信号，系统的特性是不对称的，这无论是对于建模，还是对基于模型的控制器的设计，都增加了复杂性。

为了进一步观察实际被控系统在零速度附近的输入/输出特性，向系统输入幅值为 300 数字值，周期为 10s 的正弦信号，在 5ms 的采样周期下，获得其输入/输出特性如图 9.12 所示，从图 9.12 中可以看出，在所选择的工作段，将近 40%是处于非线性摩擦力影响之下。所以一个完整的电机系统模型应当由两部分组成：在线性段工作的线性模型与在非线性段工作的非线性模型。而建立系统模型的目的，则是通过研究仿真系统的输入/输出特性，得到与实际系统性能相同，并能够代替实际系统进行控制器参数设计、调试与性能测试的可靠模型。

2） ARMAX 模型辨识

ARMAX 模型是一类广泛应用的模型结构，它可以通过重构不可观测变量估计出模型参数，并能够在估计出线性模型的同时估计出噪声模型。这更加符合系统的实际效果。

ARMAX 模型结构的形式如下：

$$A(q)=1+a_1q^{-1}+\cdots+a_{n_a}q^{-n_a}$$
$$B(q)=b_0+b_1q^{-1}+\cdots+b_{n_b}q^{-n_b}$$
$$C(q)=1+c_1q^{-1}+\cdots+c_{n_c}q^{-n_c}$$
$$A(q)y(k)=B(q)u(k-n_k)+C(q)e(k)$$

其中，q^{-1}为后向延迟因子，n_k 为延迟阶数，$e(k)$为白噪声。

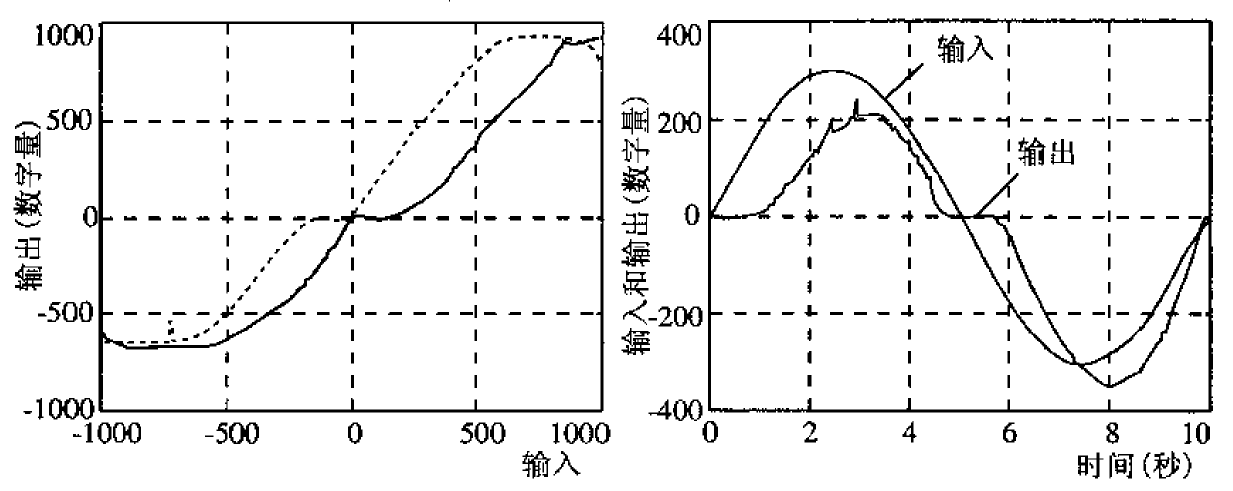


图 9.11 实际被控系统在斜坡输入作用下的上升增加（实线）和下降减少（虚线）的输入/输出特性

图 9.12 实际被控系统低速下的输入/输出特性

实验中用于线性模型辨识的输入信号为伪随机二进制序列(PRBS)，并将其加在一个直流分量上。其中 PRBS 是用来进行参数辨识的激励信号，而直流分量则是为了保证激励信号处于线性工作区。在实验中，直流分量定在 300。激励信号不能取的过小，否则会降低信噪比；也不能取的过大，否则工作区将变得过大甚至使系统进入非线性区。此外，由于系统对正负输入信号的响应具有不对称性，所以需要对被控系统的正负模型分别进行辨识。图 9.13 给出了正向辨识用输入/输出信号。

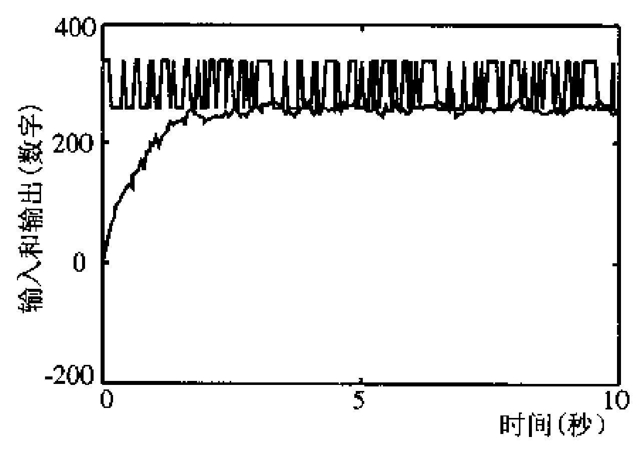


图 9.13 用于系统辨识的实际系统输入与所获系统输出的信号

离线模型的参数辨识可在辨识软件下进行。采用 MATLAB 环境下的系统辨识工具箱(SIT)。首先通过阶数辨识函数 order.m 辨识出系统的延时阶数为 $d=3$。阶数辨识函数 order.m 主要利用了 SIT 中的另外两个函数 arxstruc.m 与 selstruc.m。在辨识函数中，由于原始输入/输出信号中包含附加的直流量，可以先利用 dtrend.m 函数将直流分量去除，然后再对处理后的输入/输出数据进行辨识。辨识时所取的数据段为图 9.13 中的 700 至 1900 段的数据，通过运用通用 ARMAX 模型的辨识函数 armax.m，得到正向系统的模型为：

$$y(k)=\frac{0.0102q^{-3}}{1-0.9931q^{-1}}u(k)+\frac{1-0.9700q^{-1}}{1-0.9931q^{-1}}e(k) \tag{9.29a}$$

采用相同的方式可以得到系统的负向模型为：

$$y(k)=\frac{0.0143q^{-3}}{1-0.9910q^{-1}}u(k)+\frac{1-0.9198q^{-1}}{1-0.9910q^{-1}}e(k) \tag{9.29b}$$

9.9.2 非线性模型的建立及仿真系统的实现

因为被控系统 40%的工作区在非线性段,所以其非线性模型的建立具有更重要的意义。

一般认为，执行机构中的非线性摩擦力与速度和时间均有关系。在从静摩擦力过渡到动摩擦力的过程中，运动经历了四个阶段，有关的参数有 7 个，且参数必须通过实验确定，所以对这种非线性摩擦力模型的估计是相当复杂的。为了减少工作量，人们常常采用简化的方法，获取近似的或抓住本质特点的非线性摩擦力模型。这样，只要进行较少的实验即可确定模型的参数。笔者曾在相关的研究中提出了与速度成指数关系的非线性摩擦力的模型形式：

$$T_f(\omega)=\text{sign}(\omega)T_c+\text{sign}(\omega)(T_s-T_c)\exp^{-\alpha|\omega|} \tag{9.30}$$

其中，ω 为输出角速度；T_s 为静摩擦力矩；T_c 为库仑摩擦力矩；α 为指数时间常数；$\text{sign}(\omega)$ 为符号函数。

T_s 和 T_c 的值可以通过实验来确定，比如通过图 9.10 所获的信号，找到电机从静止到开始运动时所对应的输入值为 T_s，而从运动状态减速到静止时所对应的输入值则为 T_c。

实际系统的输出是线性模型与非线性摩擦力共同作用的结果。为了能够在计算机上重现实际系统的输入/输出特性，应当在已获得的线性模型的基础上结合非线性摩擦力的模型。在此采用(9.30)式作为非线性摩擦力的模型，并通过调整参数 α 的值，使结合后模型的输入/输出特性与实际系统的输入/输出特性相一致。

可以说，参数 α 的调节是一件耗时且依赖于经验的工作，不过在 SIMULINK 的帮助下已经可以比较容易实现 α 值的调节。线性系统模型的 SIMULINK 实现如图 9.14 所示，从图中可以看出，虽然正负方向的模型不一致，但实现起来如同写公式一样方便，噪声的实现也是相当方便的。为了简化结构，在 SIMULINK 中已用方框图将图 9.14 所示结构封装起来，作为一个子系统的模块放入。图 9.15 为带有非线性的系统模型，其中的 Linear Model With

Noise Filter 即为图 9.14 所表示的线性模型的模块。图 9.15 中其余部分为非线性摩擦力 $T_f(\omega)$ 表达式的内容。图中所看到的函数项 $f(u)$ 中的内容为指数函数。在 α 的调试过程中观察到，在电机的加速及减速过程中的 α 值也是不同的，这也使得非线性摩擦力模型更加复杂，即使如此，采用 SIMULINK 实现起来并不困难。更重要的是，通过精确的模型建立，将使得以后的控制器的设计更加符合实际系统的要求，实际上是为以后的设计打好基础。整个系统的仿真实现如图 9.16 所示，其中 Motor V_Model With Noise Filter 即为图 9.15 所表示的非线性模型的模块。另外，图中还包括用于产生输入信号的信号发生器 Signal Generator；用来观察不同点信号显示的示波器 Scope(1)；用图 9.15 具有非线性的系统模型在 MATLAB 环境下进行仿真实验时，由设计者定义的各个观察变量：输入参数

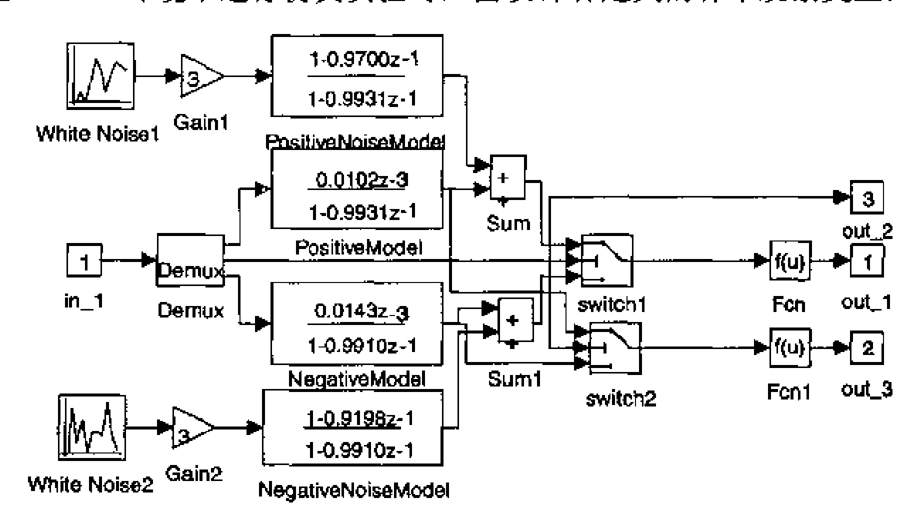


图 9.14 线性系统模型的 SIMULINK 实现图

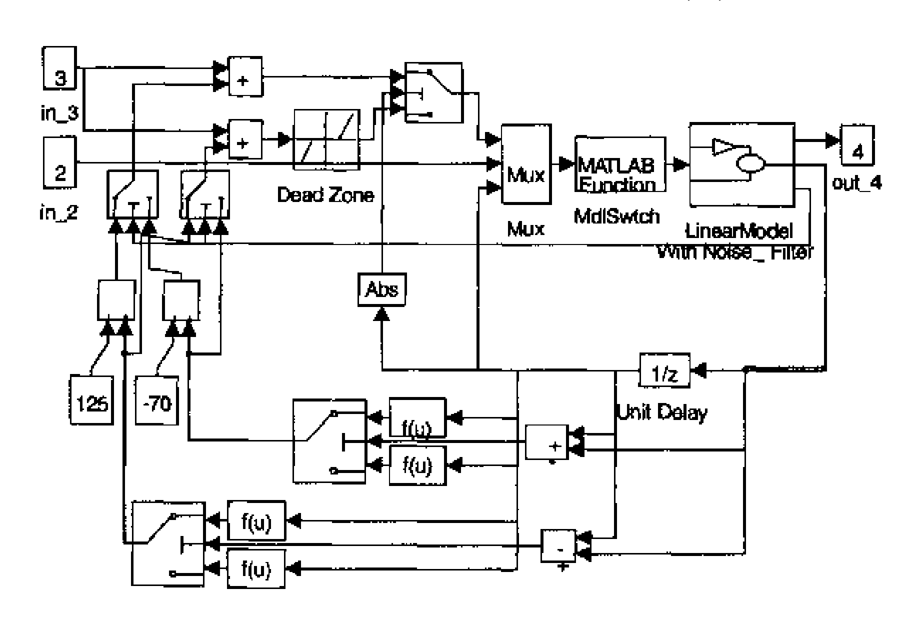


图 9.15 被控系统中非线性部分的模型

信号 R_{in}；被控输出变量 y_{out}；控制变量 U_{out} 等等。这些可以使设计者根据不同的需要在 MATLAB 环境（Workspace）下进行数据处理。控制器的设计内容将放在模块 Controller 之中。在本项工作中，Controller 中的内容仅取系统的输入，因而执行的是系统开环的测试任务。一旦模型的各个参数调试完毕，则可以通过图 9.16 所示系统进行仿真实验。

向仿真系统输入与实际系统相同的输入信号，并同时将其输出与实际系统输出进行比较，用于对比的实际系统的输出信号 y 是通过 From Workspace 功能模块获得的。最终的信号对比图如图 9.17 所示，从中可以看出实际系统与仿真系统之间的一致性。

如前所述，对线性模型的辨识是在输入信号幅值为 300 直流量上叠加幅值为 50 的伪随机二进制序列的基础上获得的，所以此种辨识方法存在着一个工作点的问题，即所建模型一般只适合于所辨识的工作点的一定范围内。为了考察在不同工作点下的仿真系统的自适应性，分别对系统输入信号频率保持不变（周期为 10 秒钟，即频率为 0.628 弧度/秒），

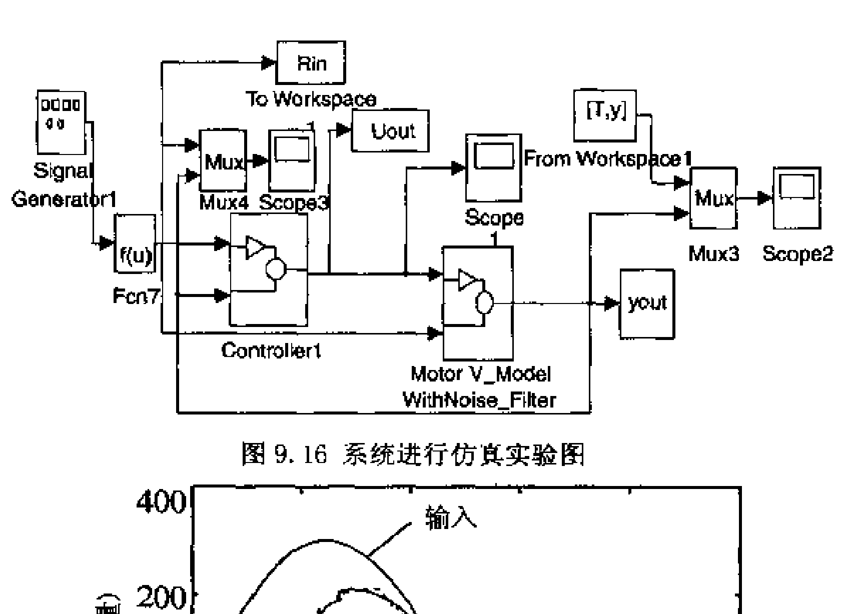


图 9.16 系统进行仿真实验图

图 9.17 输入/输出信号对比图

而幅值分别为 250、300、350 的正弦信号，以及幅值保持在 300 不变，增大或减小输入正弦信号的频率进行了对比实验。实际上，摩擦力表达式中的各参数，尤其是库仑摩擦力 T_c 对于不同幅值以及变化速度不同的输入信号所表现出的值也不尽相同。即使如此，采用(9.30)式所提出的非线性摩擦力的模型所获得的仿真系统，在所做的实际系统与所获仿真系统的对比实验中，对于输入正弦信号幅值范围区间[250, 350]内仿真系统的响应与实际系统的结果达到了相当好的一致性。

本节结合理论与实际应用的实例，详细探讨并建立了一个机电系统的非线性模型。从系统的组成、模型的选取，到 MATLAB 环境下的参数辨识，再到 SIMULINK 下的仿真过程，重点做了复杂的非线性模型的建立与实现过程的工作，并与实际系统所测信号进行了性能对比和验证。这样，为今后的控制器设计与调试，以及控制系统的仿真实验做好准备。

第 10 章 模糊控制系统的应用

经过实践，人们认识到模糊逻辑控制技术最适合用于那些非线性系统和在其输入或者对其操作描述存在着不确定性的系统。查德教授认为模糊逻辑对于大的自然系统，诸如天气、海洋系统，或者大的人造系统，诸如经济、股票市场和国家选举等这样的系统建模和控制是最有利的。用模糊逻辑去开发控制系统已经获得很大成功。一般而言，在控制应用中，模糊逻辑可以应用于非线性、时变和无法定义的系统，最适用的还是以下三类系统：

①由于太复杂而无法精确建立模型的系统；

②具有明显非线性操作的系统；

③其输入或者其定义具有结构不确定性的系统。

对于一些简单系统，有些可以知道确切的数学模型，有些甚至并不需要逻辑推理或者复杂的数据处理，控制器只要一步一步地通过预先确定的作用对输入作出简单响应，对这样的系统就不需要用模糊逻辑去实现。如果一个系统的经典控制方程和方法已经经过优化或者完全可以胜任，已经形成成熟的方法，一般情况下，没有必要因为模糊逻辑是新技术而去采用它。但是当现有方法虽可胜任，而用模糊逻辑方法可以做得更好，甚至要好得多，或者由此有可能开辟一个有意义的所需要的新功能的话，则应当采用模糊逻辑去取代它。

当今自动控制理论中一个重要的研究领域就是新的控制策略的研究与探索。自动控制应用的一个引人注目且吸引越来越多的专家投入的领域是运动控制。其中需要解决的一个重要的实际问题是：电机中非线性摩擦力的消除与补偿。在小功率电机广泛运用的今天，在跟踪低速或变向运动的参考信号时，由电机中的非线性特性所引起的误差，采用古典控制理论设计的常规控制器很难对其进行补偿与消除，控制系统的精度很难达到理想的设计要求。

为了解决运动控制应用中的非线性问题，人们不断地引入新的控制策略，从而使得控制系统更加具有智能性及抗各种干扰的鲁棒性，并且控制策略的使用与实现，不需要增加更多的硬件，只要一个以数字信号处理器为基础的计算机控制系统，从而使整个控制系统更加集成化，更加智能化，更加具有可靠性和灵活性，也更加具有广泛的实际应用价值。

智能控制系统的关键在控制策略上。一个好的控制策略可以适用于普遍的一类问题。利用模糊逻辑控制不依赖于被控对象数学模型的特点，可以设计出运动控制中具有摩擦力影响的速度模糊逻辑控制系统。通过计算机软件来实现模糊逻辑控制系统的模糊化、模糊推理和解模糊三个处理过程，并对模糊控制系统进行计算机仿真调试。最后，与常规控制器的控制效果进行比较。

10.1 速度模糊控制器的设计

模糊控制器选用二维输入和一维输出。输入变量为误差 E 和误差变化 EC，输出变量为伺服电机力矩控制量 T，误差变化在计算机中的实现是采用误差采样信号的一阶延迟。

误差和误差变化的模糊标记分别为 12 和 11 个变化值。因为误差的正零与负零对电机的转速来说是表示不同的旋转方向，分别由正负控制信号产生，所以有不同的意义。为了得到平稳地跟踪误差信号，隶属函数选用等距离交叉分配的三角形，如图 10.1 所示。

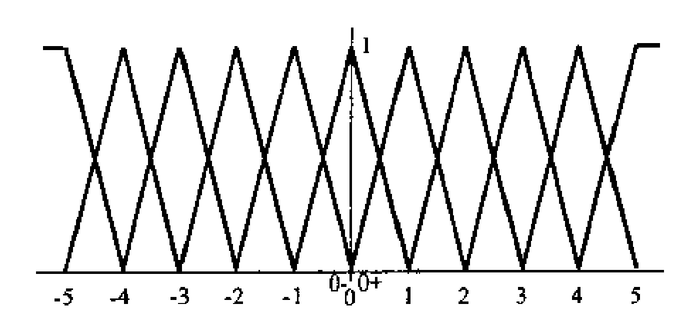


图 10.1 E和 EC的隶属函数

模糊控制规则通过计算机中程序化的算法来完成。为了计算的方便，控制规则采用带有调节因子的控制算法：

$$U=\alpha E+(1-\alpha)EC,\quad \alpha\in[0,1] \tag{10.1}$$

在通常情况下选用 $\alpha=0.5$。

控制规则含意用语言变量来描述，则形成为：

$$\text{IF } e \text{ is } E_i \text{ AND } ec \text{ is } EC_i \text{ THEN } u \text{ is } U_i \tag{10.2}$$

其中 E_i和 EC_i是定义在 E 和 EC 两前提空间的模糊集合，而 U_i是结论 U 空间影射的模糊集合，$i=1,2,\ldots,N$，N 为模糊标记的数目。

对于任意一对前提 e 和 ec，模糊算法将对每一条规则根据输入前提隶属函数最小值法得到其规则强度，可写为

$$\sigma_i=\mu_{E_i\text{AND}EC_i}(e,ec)=\min[\mu_{E_i}(e),\mu_{EC_i}(ec)] \tag{10.3}$$

根据所观察到的输入前提，模糊控制器推导出一个模糊结论 U，它的隶属函数是

$$\mu_U(u)=\max_{1\le i\le N}[\sigma_i,\mu_{U_i}(u)] \tag{10.4}$$

控制输出是对表示模糊控制信号 U 的隶属函数图形进行解模糊处理，同样为了计算的方便，本例采用单值的 I 形作为输出变量的隶属函数，采用质量中心法(COG)，结合加权平均，计算控制输出精确值。其求解公式为

$$\text{精确输出值}=\frac{\sum_i\mu_U(u)\times x}{\sum_i\mu_U(u)} \tag{10.5}$$

控制器的输入和输出均为计算机中的数字量，需要通过量化因子量化到各自的模糊标记上。从模糊控制器的调试中发现，输入变量的基本论域不仅仅表示可能涉及到的范围，更重要的是反映了控制器所能达到的精度。在速度控制系统中，当隶属函数和控制规则确定后，输入变量 e 的基本论域体现了所能达到的精度，尤其是当系统参考变量有可能在较大的范围里变化时，对于跟踪误差来说，最终目的总是希望$e\to 0$，所以应当多从精度的角度来选择误差的基本论域。至于输出变量的量化因子，可由处理器的位数来确定。如对于一个 16 位的数字信号处理器，具有 11 个模糊集合的量化因子为 $K_u=6553.6$。所有大

于或等于模糊标记 ±5 的控制量均以$2^{16}/2=32768$值输出。速度控制系统中的量化变量关系如表 10.1 所示。

表 10.1 输入输出量化变量表

量化级别	E	EC	T
+5	50	325	32768
+4	40	260	26214.4
+3	30	195	19660.8
+2	20	130	13107.2
+1	10	65	6553.6
0+	0+	0	0
0-	0-	0	0
-1	-10	-65	-6553.6
-2	-20	-130	-13107.2
-3	-30	195	-19660.8
-4	-40	260	-26214.4
-5	-50	325	-32768

被控系统具有下列开环电机械方程：

$$J\frac{\mathrm{d}\omega}{\mathrm{d}t}=-B\omega+T-T_f \tag{10.6}$$

其中，ω为角速度，J 为总惯量，B 为粘摩擦系数，T 为控制输入，T_f为非线性摩擦力矩，T_f的精确模型选为：

$$T_f=T_c\,\mathrm{sgn}(\omega)+\Delta T\mathrm{e}^{-\alpha\omega}\mathrm{sgn}(\omega) \tag{10.7}$$

为了使仿真结果更加真实地反映出精确的非线性系统特性，我们采用(9.7)式对被控系统进行仿真。仿真系统对正弦信号的输出以及所测实际被控系统对相同信号的输出如图 10.2 所示。从中还可以看出被控系统中存在着严重的非线性死区和饱和特性。

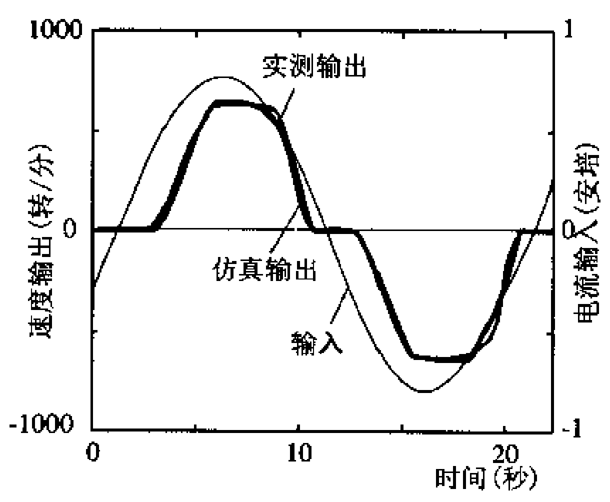


图 10.2 被控系统实测和仿真输入输出特性

被控系统中的线性部分是通过标准的 ARMAX 辨识，得其方程为

$$G(z^{-1}) = \frac{10z^{-1}}{1-0.9932z^{-1}} \tag{10.8}$$

计算机中的模糊逻辑控制仿真系统方框图如图 10.3 所示。图中电驱动器包括被控直流电机、功率放大器、信号比例调节器，以及 A/D、D/A 转换系数等整个被控子系统，S 为虚设的最大输出控制量的限定器。

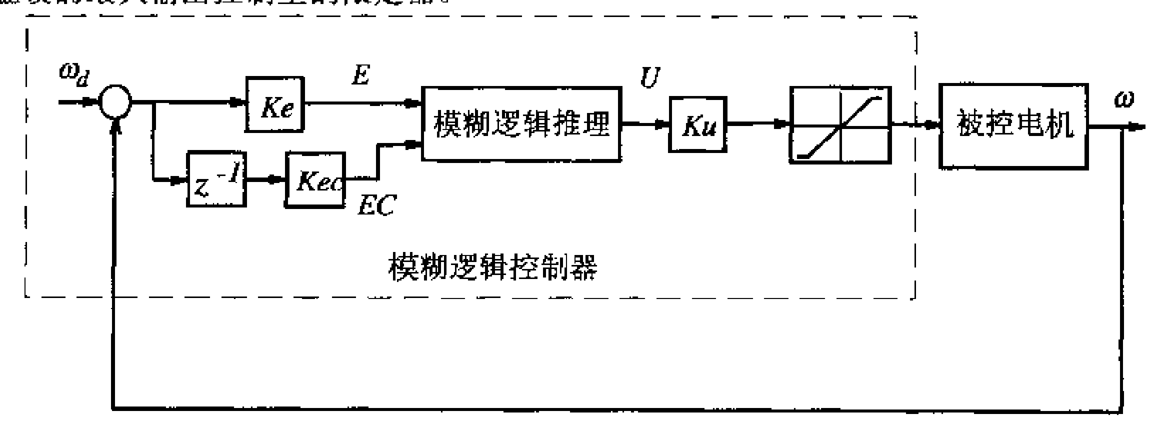


图 10.3 模糊逻辑控制仿真系统方框图

当控制规则确定后，控制系统响应的动静态性能就由 K_e、K_{ec}和 K_u 来决定。由于系统中存在着严重的死区，而正、反两个方向需要施加不同方向的力矩，为了避免控制力矩的正负振荡，允许系统有一定范围的误差，以达到平稳控制目的。为此，三个参数的选择步骤为：

①$K_u = \dfrac{\text{输出最大可达范围}}{\text{最大量化级别}}$；

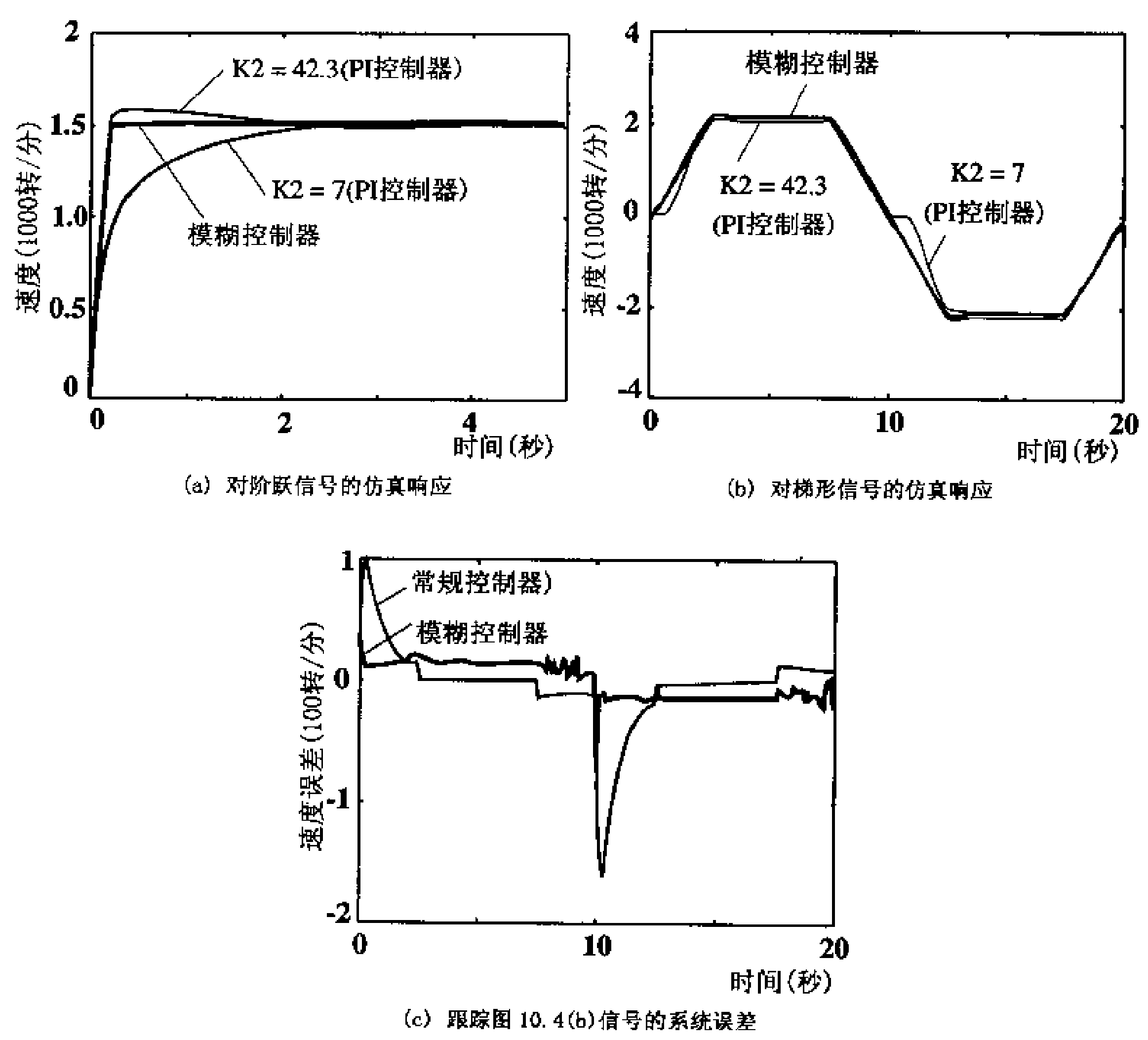
②令 $K_{ec}=0$，调定 K_e，使系统在不振荡的前提下，达到尽可能大的稳态精度；

③适当增加 K_{ec} 的值，以提高上升时间，同时又不使系统振荡。

经验表明，误差 E 的论域决定了控制器的控制精度，所以其量化系数应根据期望精度来确定。而误差变化 EC 的论域决定了上升速度，可在具体系统中根据参考量的可达范围进行调试，以达到满足跟踪精度和响应平稳要求的最佳值。

仿真实验是通过 MATLAB 支持下的 SIMULINK 实现的。常规控制器是通过极点设置求出的 PI 控制器。两者对不同参考信号的响应如图 10.4 所示。虽然 PI 控制器对速度控制系统不产生系统稳态误差，但其上升时间显然比模糊控制器长得多。模糊逻辑控制器既具有较快的上升速度，又没有响应超调。而 PI 控制器很难做到此点。当使其参数不产生超调响应时，需要较长的上升时间，更主要的是当跟踪变化的参考信号时，PI 控制器在零输入附近出现死区。消除死区的办法就是加大系统放大系数的值，但这又给系统带来超调，同样需要较长的时间达到满意的控制性能。图 10.4(a)和(b)中分别给出当 PI 控制器的比例系数 K 为 7 (阶跃响应中没有超调，但变化信号中出现死区)和 42.3 (变化信号跟踪中没有死区，但阶跃中可见超调)时模糊逻辑控制器的响应曲线。由此可见模糊逻辑控制器具有更好地适应参数变化以及抗干扰的能力。为了进一步比较跟踪误差，我们将模糊控制

器和 K 值等于 42.3 的 PI 控制器对图 10.4(b)中的梯形信号响应的误差放在图 10.4(c)中。结合图 10.4(a)我们可知，在上升速度相同的情况下，对于图 10.4(b)中相同变化的周期参考信号，PI 控制器的跟踪误差在[-150 +100] 转/分之间，而模糊控逻辑制器的跟踪误差只有 ±50 转/分。这又更加证明模糊控逻辑制器具有更好的响应性能。



(a) 对阶跃信号的仿真响应

(b) 对梯形信号的仿真响应

(c) 跟踪图 10.4(b)信号的系统误差

图 10.4 控制系统对不同输入信号的响应图

模糊控制器是一个按照人们事先设计好的控制规则对被控对象进行控制的智能控制器，而不管被控对象本身特性如何，均能很好地按照控制规则控制系统，所以它是一个非线性控制器。与 PID 控制器相比较，模糊控制器的响应速度快，超调小，且对非线性系统能够很好地进行控制，只是对速度控制有一定的稳态误差。这可以通过引入神经网络的训练来改进隶属函数以及控制规则而进一步提高控制效果。总之，当跟踪不断变化的参考信号时，利用模糊控制器来克服非线性摩擦力的影响不失为一个较 PID 为好的控制策略。

10.2 三种控制器的设计与性能比较

模糊逻辑控制对控制复杂或无法用简单数学模型表达的系统提供了一个实际的不昂贵的解决办法。基于规则的模糊控制器不需要繁琐的数学计算和复杂的数学模型，所需要的仅仅是对整体系统行为的实际理解。不论是基于数学模型所设计出的常规控制器，还是通过实验公式或经验得到的模糊控制规则，其目标都是一个，那就是希望对于参考输入产生期望的输出。本节从控制器规则本身入手，通过控制器输入输出特性表面的形状，对常规控制器、一般模糊控制器以及针对具体系统设计的非线性模糊控制器的三种控制规则进行直观地对比，以展现各控制器之间的区别，以及非线性模糊控制规则的实质。

10.2.1 控制算法的描述

1） 常规的线性控制器

众所周知，PD 控制器的控制算法可以用下式表示：

$$u(k) = K_1 e(k) + K_2 e(k-1) \tag{10.9}$$

此处，$u(k)$为第 k 个采样周期的控制量；$e(k)$和 $e(k\text{-}1)$分别是误差信号在第 k 和 $k\text{-}1$ 采样周期的参考信号与系统实际输出值之间的差值；K_1 和 K_2 为比例常数。

因为(10.9)式是个线性方程，此控制器的特性表面是一个平面（即输出控制信号 $u(k)$ 与输入信号 $e(k)$和 $e(k\text{-}1)$关系的立体空间图形）。这个表面如图 10.5 所示。

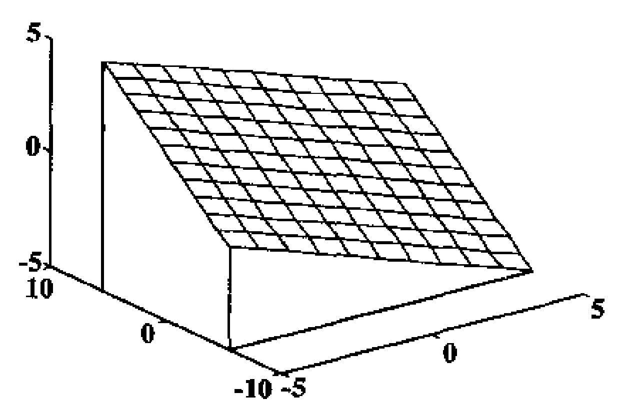


图 10.5 PD 控制器特性表面图

2) 模糊控制器

一般模糊控制器的输入 e、ec 和输出 u 的基本论域取：$[-E_{max}, E_{max}]$、$[-EC_{max}, EC_{max}]$ 和$[-U_{max}, U_{max}]$，其模糊论域分别选用比如 12 和 11 个模糊集合，这样可得 132 条控制规则。量化因子可选为 K_e=5/E_{max}，K_{ec}=5/EC_{max} 和 K_u=U_{max}/5。输入变量的隶属函数采用常用的等距离交叉分配的三角形，如图 10.6 所示。

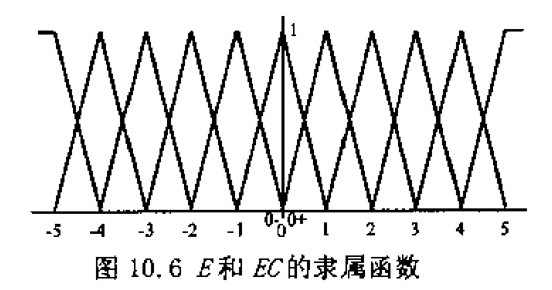


图 10.6 E和 EC的隶属函数

输出控制变量采用单值的 I 形隶属函数，通过质量中心法(COG)，结合加权平均，可得控制输出的精确值，其公式为：

$$\text{精确输出值} = \frac{\sum_i \mu_u(u)\times x}{\sum_i \mu_u(u)} \tag{10.10}$$

此处，μ_u 为对应于输入变量的输出模糊标记的隶属度；x 为各模糊标记值。

3）非线性模糊控制规则(NFLC)的设计

在此设计两种不同的控制规则。第一种如表 10.2 所示。这是人们常用的带有调节因子 $u=\text{int}(\frac{e+ec}{2})$ 的控制规则。

表 10.2　PD 型模糊控制规则

	-5	-4	-3	-2	-1	0	1	2	3	4	5
-5	-5	-5	-4	-4	-3	-3	-2	-2	-1	-1	0
-4	-5	-4	-4	-3	-3	-2	-2	-1	-1	0	1
-3	-4	-4	-3	-3	-2	-2	-1	-1	0	1	1
-2	-4	-3	-3	-2	-2	-1	-1	0	1	1	2
-1	-3	-3	-2	-2	-1	-1	0	1	1	2	2
-0	-3	-2	-2	-1	-1	0	1	1	2	2	3
+0	-3	-2	-2	-1	-1	0	1	1	2	2	3
1	-2	-2	-1	-1	0	1	1	2	2	3	3
2	-2	-1	-1	0	1	1	2	2	3	3	4
3	-1	-1	0	1	1	2	2	3	3	4	4
4	-1	0	1	1	2	2	3	3	4	4	5
5	0	1	1	2	2	3	3	4	4	5	5

按照表 10.2 所得到的控制器的误差输入 e 和 ec 与控制输出 u 之间的特性表面如图 10.7

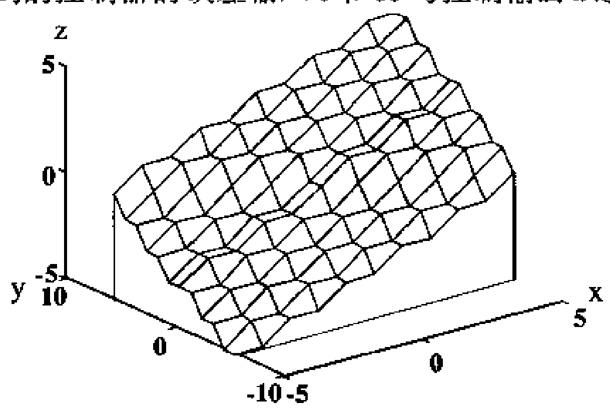


图 10.7 PD 型模糊控制规则输入输出特性表面

所示，其中，x、y 和 z 轴分别代表输入输出变量 e、ec 和 u。此控制器的特性虽然具有非线性，但仍然为 PD 输出特性（见图 10.7），我们称之为 PD 型模糊控制规则。

第二种控制规则是在第一种的基础上进行一些调整，以改进系统对阶跃响应的过渡过程状态。其调整的主要目的是在初始阶段以及达到稳态时，加大误差的影响，以至于当响应开始以及达到稳态时，能分别有更快的加速度和减速度。我们称此为非线性模糊控制规则，如表 10.3 所示。

表 10.3 非线性模糊控制规则

	-5	-4	-3	-2	-1	0	1	2	3	4	5
-5	-5	-5	-5	-5	-5	-5	-4	-4	-4	-4	-4
-4	-4	-4	-4	-4	-4	-4	-4	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-2	-2	-2
-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-0	-1	-1	-1	0	0	0	0	0	1	1	1
+0	-1	-1	-1	0	0	0	0	0	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2	2	2	2
3	2	2	2	3	3	3	3	3	3	3	3
4	3	3	3	3	4	4	4	4	4	4	4
5	4	4	4	4	4	5	5	5	5	5	5

非线性模糊控制规则与 PD 型控制规则的主要区别在于“如果 e 是+5 且 ec 是-5”，其控制输出 u 现在是“+4”，而不是“0”。这样，在响应的初始阶段，我们忽略了 ec 的影响。非线性模糊控制器的其他规则，是在不同的模糊标记之间平滑地改变其控制信号，另外“如果 e 是 0 且 ec 是 0，那么 u 是 0”，以消除稳态常数误差。非线性模糊控制规则的特性表面如图 10.8 所示，从中可以看出，规则在 e 为 0 左右有一个变化点以适应非线性控制的要求。

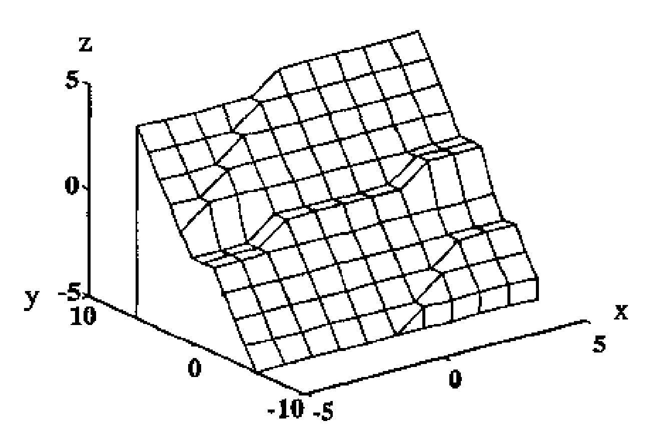


图 10.8 非线性模糊控制规则输入输出特性表面

10.2.2 结果的对比

对三种不同的控制规则均进行了仿真实验测试和结果的比较。被控系统为一个存在着严重的非线性摩擦力影响的直流电机，其位置控制系统的线性方程为：

$$G_p = \frac{0.2z^{-1}}{(1-0.9928z^{-1})(1-z^{-1})} \tag{10.11}$$

非线性摩擦力模型通过实验，确定为指数型函数：

$$T_f=\begin{cases}0.25+0.09\mathrm{e}^{-0.0024\omega}, & \omega>0\\ 0.23+0.09\mathrm{e}^{-0.0024\omega}, & \omega<0\end{cases} \tag{10.12}$$

仿真实验是通过 MATLAB 环境下的 Simulink 进行的。采样时间为 5 毫秒。对阶跃信号的跟踪响应如图 10.9 所示，其中，PD 控制器是采用极点配置方法设计的。虽然是对一个非线性系统进行控制，常规的 PD 控制器仍能表现出较好的控制特性，这是因为二阶位置系统本身带有一个积分器，能够消除库仑摩擦力的影响。虽然系统此时存在一定的超调，但具有较快的上升时间和较小的振荡。

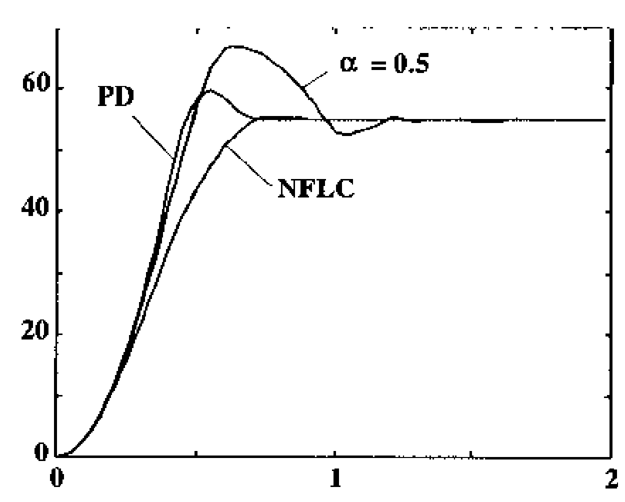


图 10.9 三种控制规则对位置跟踪控制系统的阶跃响应

带调节因子且 $\alpha=0.5$ 的模糊控制器在对具有非线性扰动的二阶系统进行控制时，并没有表现出较常规控制器更好的特性。虽然经过努力调节其他可调因子，但最后结果仍然具有较 PD 控制器大得多的超调量和更长的过渡过程。对系统性能的进一步改善没有起到积极的作用。这说明$\alpha=0.5$ 的模糊控制规则不是对任何系统都能达到比常规控制器更好的效果。

图 10.9 中的非线性模糊控制器(NFLC)对阶跃信号的跟踪，给出了一条近似对一阶系统的响应曲线，系统的跟踪既平稳又快速地达到稳态信号，这说明针对具体系统设计的非线性模糊控制器较好地克服了非线性摩擦力的影响，因而表现出较 PD 控制器更好的控制效果。

模糊控制器能根据专家的经验对非线性系统进行有效的控制。实际中的模糊控制规则常常采用带有调节因子的控制规则，但 $u=0.5(e+ec)$ 的控制规则在二阶位置控制系统中呈现出和 PD 控制器相似、甚至不如常规控制器的控制效果。而根据具体系统的非线性特性设计出的非线性模糊控制规则，通过对具有非线性摩擦力影响的直流电机的仿真实验可看出，此控制器对阶跃参考信号的响应比常规控制器有更好的控制效果。不过，由于影响模糊逻辑控制器的参考因素甚多，只凭经验或人工调整参数达到最优模糊控制既费时，又困难。改进的办法则是应用神经网络自学习自适应能力来自动调整隶属函数和控制规则，以达到最佳设计。

10.3 变参数双模糊控制器

在小功率电机越来越得到广泛应用的今天，人们更加重视对电机中非线性摩擦力的消除与补偿的研究。采用经典控制理论设计的常规控制器，在实际应用中对变速或变向的参考信号进行跟踪时，往往无法达到令人满意的效果。与常规PID控制器相比，模糊控制器对阶跃响应具有上升速度较快，过渡过程时间较短，对参数变化不敏感等优点，同时不需要被控对象的精确模型，这无疑有助于对电机的非线性的消除。但它对非线性的消除是以降低系统稳态精度为代价的。所以单个模糊控制器在非线性补偿上受到了限制。通过前面的研究表明，直流电机中的非线性摩擦力矩模型可以由下面的公式表示：

$$T_f = T_c \cdot \mathrm{sgn}(\omega) + \Delta T \cdot \mathrm{e}^{-\alpha\omega} \cdot \mathrm{sgn}(\omega) \tag{10.13}$$

这是一个以电机角速度为自变量的指数衰减型非线性函数。根据此公式，我们可以认为，直流电机中非线性摩擦力的影响只作用于速度为零附近的一个范围$|\omega| < \omega_0$内，在此之外，干扰 T_f则近似为一个常数 T_c。为此，我们设计了一个基于经验的变参数双模糊控制器，其控制转换开关在$|\omega| = \omega_0$处。其主要思想表现在两点上：

① 在$|\omega| < \omega_0$时，模糊控制器的主要目的是非线性影响的消除；

② 在$|\omega| \geq \omega_0$时，控制的重点则转移到系统跟踪精度的提高上。

此控制器的最大特点为：采用很少的模糊标记和简单的设计方法，并以ω_0为分界点来进行变参数双模糊控制，从而达到比常规控制器和单模糊控制器更好的控制效果。

在下面的几节里，首先介绍变参数双模糊控制器的设计过程，重点在于解释如何确定两组变化的参数值，然后对所设计的双模糊控制器进行计算机仿真实验的验证，并将其结果与常规控制器以及单模糊控制器的控制效果进行比较，最后给出小结。

10.3.1 参数的设定

1）模糊逻辑设计

我们以误差E和其变化值EC作为模糊控制器的输入，以伺服电动机力矩控制量U作为控制器的输出。它们的论域均以NL(负大)、NS(负小)、ZO(零)、PS(正小)和PL(正大)这五个语言变量来描述，并且将它们分别定义在-2，-1，0，1和2附近。控制规则采用带调节因子α的算法，即：

$$U = \alpha \cdot E + (1-\alpha) \cdot EC, \quad \alpha \in [0,1] \tag{10.14}$$

在实际的仿真实验中，我们得出$\alpha = 0.5$是较好的取值，所以控制规则有更简单的表达式：

$$u = \frac{1}{2}(E + EC) \tag{10.15}$$

E 和 EC的隶属函数选用等距离交叉分配的三角形，如图 10.10 所示。

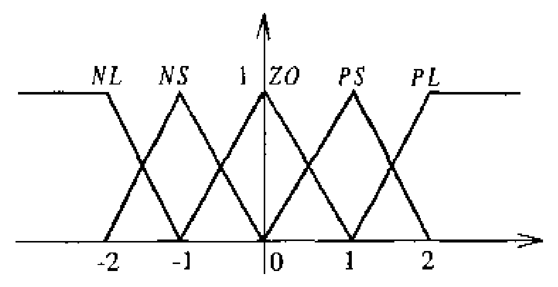


图 10.10 误差 E 和其变化 EC 的隶属函数

为了使控制算法简单化，其模糊推理方式采用简单的极小-加法-质心推理法。具体的做法和 10.1 节采用的完全相同，即每一规则的规则强度根据输入前提隶属函数最小值法得到：

$$\text{if } e \text{ is } E_i \text{ and } ec \text{ is } EC_i \text{ then } u \text{ is } U_i \tag{10.16}$$

控制量 U 的隶属函数则为：

$$\mu_U(u) = \sum_{1 \le i \le N} [\sigma_i, \mu_U(u_i)] \tag{10.17}$$

解模糊采用单值的 1 形作为输出变量的隶属函数，利用质心中心法(COG)，结合加权平均，计算其精确值：

$$\text{精确输出值} = \frac{\sum_i \mu_U(u_i) \cdot u_i}{\mu_U(u_i)} \tag{10.18}$$

2) 两组参数的设定

模糊控制器是根据操作者的经验而设计，所以对所要解决问题的准确认识和深入分析是设计成功的关键所在。在隶属函数与控制规则已定的情况下，其性能由 K_u，K_e 和 K_{ec} 这三个量化参数来决定。又因为控制力矩 T 的大小直接与 Ku 相关，故参数设定从 Ku 入手。

当变化的参考信号 $|\omega| < \omega_0$ 时，系统处于非线性摩擦力 T_f 的影响区间。此时因为输入信号较小，控制力矩的大部分用于消除非线性摩擦力矩上，只有小部分用来控制其运动。由于非线性摩擦力矩是以指数型函数迅速衰减的，如果 T 过大，则随着需要补偿的力矩的减小，会使输出信号过大而产生振荡，甚至造成系统的不稳定。所以 T 值要适中，即 K_u 要适中，以刚好抵消常数摩擦力矩值为最佳量化因子值。当然，由于 K_u 值的限制，系统的稳态精度也同时受到了限制。确定了 K_u 之后，在使系统稳定的前提下，尽可能地取较大的 K_{ec} 值来获得较高的控制精度。

当 $|\omega| \ge \omega_0$ 时，系统进入线性区域，此时有 $T_f \approx T_c$，控制力矩只要在克服常数摩擦力矩 T_c 之后，再施加有规则的控制量即可。所以此时可以在保证系统稳定的前提下，尽可能地加大 K_u 来达到提高精度的目的。

在作了如上分析之后，速度控制系统中的输入输出量化变量关系的设计如表 10.4 和表 10.5 所示。表中的输入输出变量值为计算机中所用的数字值。

表 10.4 $|\omega| < \omega_0$ 时模糊控制器输入输出量化变量关系

量化级别	e	ec	T
+2	125	1000	24000
+1	62.5	500	12000
0	0	0	0
-1	-62.5	-500	-12000
-2	-125	-1000	-24000

表 10.5 $|\omega| \geq \omega_0$ 时模糊控制器输入输出量化变量关系

量化级别	e	ec	T
+2	125	333	50000
+1	62.5	167	25000
0	0	0	0
-1	-62.5	-167	-25000
-2	-125	-333	-50000

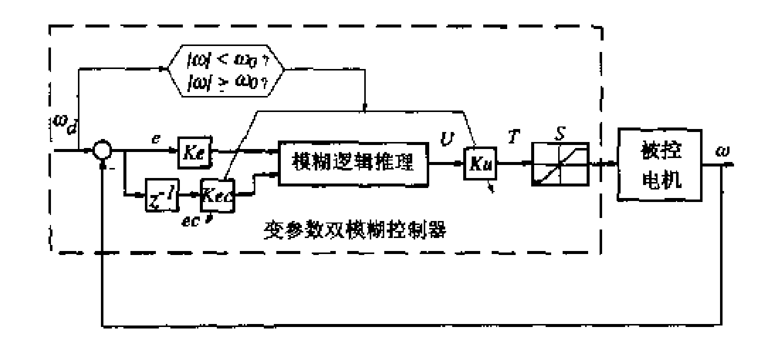


图 10.11 变参数双模糊控制器的系统方框图

10.3.2 仿真实验验证

为了验证所设计的变参数双模糊控制器的控制效果，我们在 MATLAB 环境下用 Simulink 进行仿真实验，系统的采样时间为 5 毫秒，这与实际计算机控制系统的情况是相符的。仿真系统方框图如图 10.11。

系统图中的变参数双模糊控制器是按前面的方案设计的。S 为虚设的输出控制量的最大值限定器。电驱动器部分包括被控直流电机、功率放大器、信号比例调节器及 A/D、D/A 转换等整个被控子系统。

速度控制系统的线性方程的 Z 域传递函数与(10.8)相同，为：

$$G(z^{-1})=\frac{10\cdot z^{-1}}{1-0.9932\cdot z^{-1}} \tag{10.19}$$

系统非线性摩擦力矩的模型仍然取(9.11)式，为：

$$T_f=\begin{cases}0.25+0.09\mathrm{e}^{-0.0024\omega}, & \omega>0\\ -0.23-0.09\mathrm{e}^{-0.0024\omega}, & \omega<0\end{cases} \tag{10.20}$$

输入参考信号采用图 10.12 所示的梯形波。在整个信号的跟踪过程中，采用表 10.4 和表 10.5 所设定参数的双模糊控制器参数。ω_0 取 40 转/分钟。为了便于比较，我们同时对系统进行了 PI 控制器的实验。图 10.13 分别为变参数双模糊控制器和 PI 控制器对图 10.12 输入信号的响应误差曲线。可以很明显地看出，虽然 PI 控制器对常数干扰不产生误差，但在跟踪穿越零的变向速度信号时，由于静动摩擦力的影响而产生突变的跳跃。与其相比，变参数双模糊控制器具有令人满意的控制效果。它根据被控系统的特点，各用一组参数分别侧重解决问题的一个方面，做到了结合两个模糊控制器单独使用时的优点，同时回避了各自的缺点，从而达到了较好的控制效果。在两个控制器之间相互转换的过程中没有出现大的误差跳跃，这也是单模糊控制器与 PI 控制器相结合所不能达到的效果。

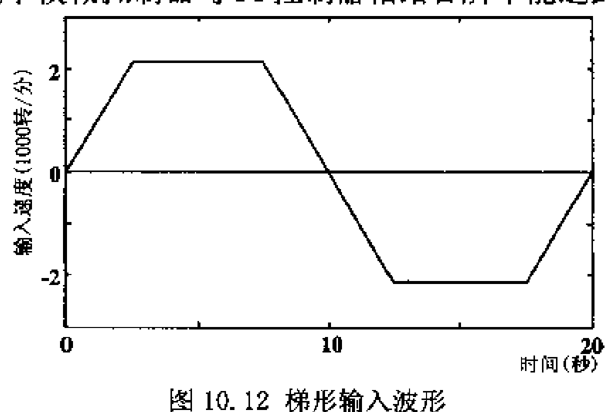


图 10.12 梯形输入波形

图 10.13 变参数双模糊控制器与常规控制器的响应误差

为了更深入地剖析变参数双模糊控制器的特点，我们将其与单模糊控制器使用时的效果进行比较，其响应误差如图 10.14 所示。从中可以看出，两者的差别仅在信号为零时刻。

单模糊控制器虽有较强的适应参数变化的鲁棒性，但由于要在系统稳定条件下兼顾全范围的跟踪精度，不得不放弃对非线性进一步的补偿而保证一定的稳态跟踪精度。变参数双模糊控制器可以在保证系统稳定的前提下进一步对系统中的非线性进行补偿。这就克服了单模糊控制器的不足，使变参数双模糊控制器比单模糊控制器的性能更好。

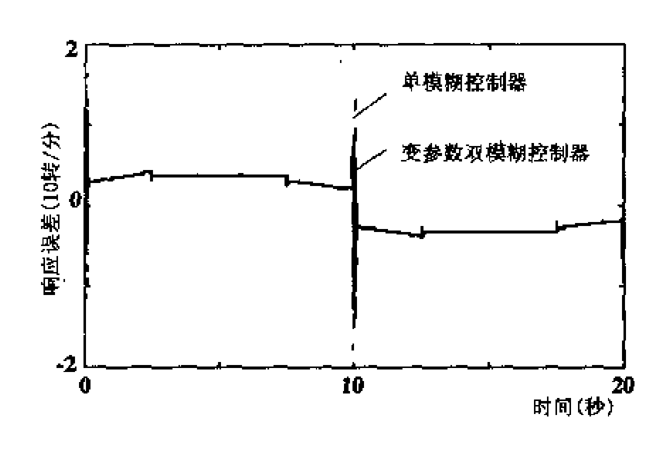


图 10.14 双模糊控制器和单模糊控制器的响应误差

10.3.3 小结

电机中的非线性是由指数型的静动摩擦力造成的。对它的消除与补偿，从理论上讲可以通过在线性控制量上加一个相反方向且幅值相等的摩擦力即可。但由于自身的非线性，采用常规控制器很难将其预测并加以消除。单模糊控制器对参数变化有较强的鲁棒性，已能较好地补偿系统中的非线性，但由于系统稳定与跟踪精度之间的矛盾，从而限制了对非线性的进一步补偿。本节所提出的变参数双模糊控制器设计方法简单，仅用 5 个语言变量就达到了其他模糊控制器采用 11 个语言变量时的控制效果。能够很好地兼顾摩擦力补偿与稳态精度两方面的要求，发挥了模糊控制器的特长，分段利用了两个模糊控制器单独使用时的优点，取得了比常规控制器及单模糊控制器更好的控制效果。此外，本节提出的变参数双模糊控制器，可以扩展到变参数多模糊控制器，用来解决其他非线性控制问题。

第 11 章 神经网络的应用

11.1 BP 网络结构、参数及训练方法的设计与选择

前向 BP 网络以其优良的非线性逼近性能，受到各个领域人们的关注。应用神经网络的关键在于网络的结构与参数的设计上，而网络结构与参数的设计，首先取决于对人工神经网络理论的理解和掌握，然后取决于设计网络的实际经验。因为在网络设计过程中，涉及到较多参数的确定。例如网络的层数、每层的神经元节点数、初始值的选取、学习速率的确定，最重要的还有训练算法的选定或改进。这些对于初次进行网络设计的人来说，往往可能无从下手。本节的目的在于根据已有的 BP 网络设计及其改进方案，对一个人工神经网络进行具体的设计，通过其详细的设计步骤与过程，对网络隐含层神经元数、初始权值、学习速率等参数在 BP 网络设计过程中的关系与影响，以及不同的改进算法在网络训练中所起的作用给予进一步揭示，使人们从中得到更多的启迪，以便使更多的人能够设计出效率更高、精度更好的神经网络。

一般情况下，一个 BP 网络的设计应当考虑以下几个方面：

①网络结构，包括网络层数、每层的节点数及其激活函数的类型；

②训练前初始参数的选取，包括：a)初始权值，b)（初始）学习速率，c)（自适应时）递增（减）因子的选取，d)最大训练次数；

③权值的训练方法；

④各种参数的先后调节（选取）次序；

⑤训练程序的编写；

⑥网络的最后选定（训练过程中不同参数情况下的性能对比研究）。

以上所有的方面，必须经过上机训练，经过多次的调整与对比试验才能设计出一个满意的 BP 网络。所以，只通过书本，看懂 BP 网络的设计原理，训练权值的计算公式和设计过程的人不能说自己已掌握了 BP 网络的设计，必须真正自己动手，在考虑并认真去实施上述的各步骤后，设计出一个达到满意精度的网络后，方可说已经有了些设计 BP 网络的经验。

下面就上述设计步骤中的各种考虑，给出一般在网络设计过程中的具体考虑。

11.1.1 BP 网络的设计

人工神经网络是由人工神经元的并联和 / 或串联组合而成的。一个人工神经元是由多输入节点和单输出节点之间通过某种激活函数的相互联接所构成的单层网络。BP 网络是采用误差反向传播算法对网络权值进行训练的多层前向网络。BP 网络设计的最大特点就

是网络的权值是通过使网络输出与样本输出之间的误差平方和达到期望值而不断调整网络的权值训练出来的。为此，训练网络的算法是能否达到训练目标的关键。BP 算法是每个初学者都必须掌握的训练算法。但掌握了 BP 算法是否就能设计出满意的网络？回答往往是否定的。因为 BP 网络的设计过程是一个参数不断调整的过程，这意味着是一个不断对比结果的过程。所以此过程是比较复杂和带有经验性的。学过 BP 网络理论设计的人都已知道，在 BP 网络的设计中，以上所提及的都是必须考虑的方面。但是如何考虑？先后顺序是什么？对于不同的设计者，由于理解和掌握的不同，经验不同，对同一问题所设计出的网络，在网络结构与效果上都可能会有很大的不同。

1）网络结构

进行网络设计的首要任务就是网络结构的确定，这包括：输入／输出节点、层数、每层的激活函数的确定以及隐含层节点数。

（1）输入／输出节点

输入／输出节点是与样本直接相关的。BP 网络已被应用于各个领域，无论让它完成什么任务，都必须将实际问题转化为网络能够接受的形式——数据样本。如果样本格式已确定，则网络的输入／输出节点数由样本固定。实际上，在大多实际应用中，是需要设计者根据实际情况来确定样本的，此时，输入／输出节点数的确定也是需要认真考虑的，尤其是在建模中，如何设计出一个能够表现出被控过程动态特性的模型是需要相关的专业知识的。一般情况下，输入／输出节点的设计是需要有关于网络实现任务的专业方面的知识。网络实现效果不好的绝大部分原因，是因为设计者未对所要实现任务的输入／输出变量（节点）数量以及各个变量之间的相互关式准确掌握所致。而这一点是网络设计之外的功夫与知识。

(2) 层数

BP 网络所具有的最大也是唯一的特点是非线性函数的逼近，而且只含有一个隐含层的 BP 网络即可完成此任务。由于 BP 网络的功能实际上是通过网络输入到网络输出的计算来完成的，所以多于一个隐含层的 BP 网络虽然具有更快的训练速度，但在实际应用中需要较多的计算时间。另一方面，众所周知，训练速度也可以通过增加隐含层节点数以及采用更好的训练算法来达到。所以，从实用的角度出发，除有特殊的要求外，对于一般的应用情况，采用具有一个隐含层的 BP 网络就能够达到目的。

（3）每层激活函数

众所周知，BP 网络的非线性逼近能力是通过 S 型的激活函数体现出来的。所以，隐含层中一般采用 S 型的激活函数，输出层的激活函数可以采用线性或 S 型。当希望网络的输出范围是$(-\infty, +\infty)$时，应当采用线性激活函数。当采用 S 型的激活函数作为输出层的激活函数时，其非线性逼近速度快于线性激活函数，但此时的网络输出被限制在（0，1）或（-1, 1）之间。

2) 各个初始参数的取值

在网络初始参数的确定上，最主要的应当是隐含层节点数 S1 的确定，但这又是一个几乎让所有初学者困惑的事情。实际上，隐含层节点数应当根据具体问题，通过实验确定。

作为网络设计的例题，在此选择一个简单的控制器设计问题：设计实现模糊控制规则

为 $U=\text{int}\frac{1}{2}(e+ec)$ 的模糊神经网络控制器，此处我们定义取整函数 $\text{int}(x)$ 为不大于 x 的最大整数。

（1） 输入/输出数据对的确定

①网络由两个输入变量 e（误差）和 ec（误差的变化）构成输入向量 P = [e;ec]；

②期望目标 $T=\text{int}\frac{1}{2}(e+ec)$。

对于由输入变量 e 和 ec 以及期望的输出目标 T，由此可以得到采用神经网络实现其控制规则的输入和输出向量为：

```
P =[-2 -1 0 1 2 -2 -1 0 1 2 -2 -1 0 1 2 -2 -1 0 1 2 -2 -1 0 1 2;
    -2 -2 -2 -2 -2 -1 -1 -1 -1 -1 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2];
T = [-2 -2 -1 -1 0 -2 -1 -1 0 0 -1 -1 0 0 1 -1 0 0 1 1 0 0 1 1 2];
```

（2） 网络的结构

根据以上分析，我们采用 2 - S1 - 1 的网络结构：在隐含层采用 S 型激活函数，输出层采用线性激活函数，隐含层中神经元的个数 S1 待定。

（3）隐含层节点数 S1 的确定

在网络初步确定结构后，为了能够通过对网络的训练来确定 S1 值，还必须在训练前确定各个初始参数取值。

一般而言，学习速率 lr 对训练结果的影响在于当 lr 过大时，可能导致系统不稳定，过小时会导致较长的训练时间，一般 lr 取值在 0.01 到 0.8 之间。为了避免误差落入的是误差曲面的局部最小值，并显示各种不同算法的优劣性，我们确定最大训练次数为 10000 次，并固定目标函数(均方误差)为 0.001 来进行实验，实验中使用的 MATLAB 版本为 5.2，实验环境是 PC-PIII500。

取最大训练循环次数 max_epoch = 10000；目标误差 err_goal = 0.001。此时，所有参数均不确定，故只有采取尝试的办法，通过不断实验来逼近，为了保证一定的收敛速度，将 lr 取得稍大，通过训练记录数据见表 11.1。

表 11.1 取 lr = 0.020 时的网络训练记录

S1	误差平方和	训练次数	时间(秒)	误差过程记录
6	0.6550	10000	272.4300	有收敛趋势
7	0.0412	10000	263.6400	有收敛趋势
8	0.0826	10000	264.1300	7800 次后出现毛刺现象，但不影响收敛趋势
9	0.2633	10000	266.2200	2000 次后开始振荡，4000 次后开始发散

由表 1.11 可以看出训练效果不好。推测是 lr 取值过大所致，因而使得系统稳定性不好，故减小 lr 的取值，重新获得训练结果见表 11.2。

由表 11.2 可以看出:S1 的增加能加速误差的下降，但是随着 S1 的增加，每一次循环过程中进行的计算量也随之增加，所以所需要的时间不一定随之减少。由于在训练中，不同 S1 情况下的初始权值[W10,B10]，[W20,B20]均是随机的，故对 S1 的性能比较有一定的影响，但我们仍可以看出整体趋势：当 S1 = 7，8，9，10，11，12 和 15 时，都是能够解决问题的且误差减少速率是越来越快的，不过从网络实现的角度上来看，S1 取得越大，网络实现中的计算就越费时。根据 S1 选择在能够解决问题的前提下适当加一点余量以加

快误差下降速度的原则，根据表 11.1 和表 11.2 所示的具体情况，选取 S1 = 9。

表 11.2 取 lr = 0.015 时的网络训练记录

S1	误差平方和	训练次数	时间(秒)	误差记录过程
4	0.7696	10000	276.0600	平滑曲线，有向下收敛趋势
5	1.0821	10000	268.1500	500-800 间有毛刺
6	0.5800	10000	267.3200	开始明显下降，但 5500 次后有小毛刺
7	0.0968	10000	263.9700	下降趋势明显，曲线平滑
8	0.0010	9182	261.5000	快速下降，曲线平滑
9	*0.0010*	*7131*	*189.3900*	平滑快速下降
10	0.0235	10000	274.1900	明显较快平滑下降曲线
11	0.1169	10000	267.6500	1800 次后出现毛刺，有下降趋势
12	0.0010	4663	125.3400	很快的平滑下降
15	0.0010	1761	48.2200	下降很快，但初始误差非常大，且开始有峰

3) 初始权值的选取

一般取初始权值在(-1,1)之间的随机数。另外，威得罗等人在分析了两层网络是如何对一个函数进行训练后，提出一种选定初始权值的策略：选择权值的量级为$\sqrt[r]{s1}$，其中 $s1$ 为第一层神经元数目。利用他们的方法可以在较少的训练次数下得到满意的训练结果。在 MATLAB 工具箱中可采用函数 nwlog.m 或 nwtan.m 来初始化隐含层权值 W1 和 B1。其方法仅需要使用在第一隐含层的初始值的选取上，后面层的初始值仍然采用随机取数。在本节所给的例题中，确定初始权值如下：

```
[W10,B10]=nwtan(S1,R); [W20,B20]=rands(S2,S1);
```

我们在每次确定的 S1 下进行 5 次训练，从中选择训练结果最好的一次的[W10,B10]和[W20,B20]，作为网络的固定权值来进行后面的对比实验。

在 S1 = 9，lr = 0.015 下的 5 次随机实验中最好的一次结果为：

误差平方和（SSE）= 0.00099989；
训练次数（epochs） = 7131；
时间（time） = 189.3900(秒)。

此时对应的初始权值为：

```
W10 =[2.2312 -2.2035;1.1189 -0.6776;1.5390 -1.7795;
      1.3675 1.3787;1.6230 1.4963;-0.9914 -1.9675;
      1.7568 2.3563;-1.3277 1.1191;2.9429 2.7832];
B10=[-1.9254;-0.2765;2.0183;2.6543;-0.7643;1.0097;-1.7705;-1.6106;2.5937];
W20 =[-1.3604 1.8750 -0.7371 -1.2319 1.7746 0.7099 -0.8243 -0.8762 1.3169];
B20 =[-0.1761];
```

实验中发现初始权值对训练结果的影响至关重要，即使是很微小的对初始权值的改变，也会带来误差记录的剧烈变化，表现为出现毛刺、不平稳或下降趋势改变，甚至出现发散现象。

4）学习速率的影响

为了进行对比，在[W10,B10],[W20,B20],S1,max_epoch 和 err_goal 给定情况下，改变学习速率 lr 的值，分别对网络进行重新训练，获得记录如表 11.3 所示。

表 11.3 不同学习速率时的网络训练结果

Lr	SEE	Epochs	Time(s)	误差记录过程
0.005	0.0026	10000	269.6300	平稳下降曲线
0.010	0.0011	10000	287.7500	平稳下降曲线
0.012	0.00099986	8945	277.3700	平滑快速下降曲线
0.015	0.00099989	7131	189.3900	平滑快速下降曲线
0.0151	*0.0009982*	*7083*	*188.9400*	平滑快速下降曲线
0.0152	0.00099988	7035	201.4100	平滑快速下降曲线
0.0153	0.00099999	7913	244.6400	开始下降很快，6600 次后出现大尖刺
0.016	0.0086	10000	323.1300	4500 次开始有振荡感毛刺但仍为下降
0.0275	0.1345	10000	268.5300	很多毛刺，趋向平直
0.0280	--------	-----	---------	发散

由表 11.3 可以看出：较大的学习速率在训练的初始阶段能加速误差减少，但随着训练的不断深入，由于学习速率过大，使网络每一次的修正值过大，而导致在权值的修正过程中超出误差的最小值而永不收敛；另外较大的学习速率也容易引起振荡而难以达到期望目标。对照表 11.3 所示的各方面因素，可取 lr = 0.0151。

11.1.2 采用自适应学习速率与固定学习速率的比较

为了对不同改进算法的效果也进行对比，特采用自适应学习速率的算法，重新对网络进行训练，此时取初始学习速率值 lr = 0.020；递增乘因子 lr_inc = 1.05；递减乘因子 lr_dec = 0.70；误差速率 err_ratio = 1.04。得到：

SSE = 0.00099753；

epochs = 6728；

time = 178.1800(s)。

与表 11.3 中固定学习速率中最好的情况相比：

lr = 0.0151；

SSE = 0.00099982；

epochs = 7083；

time = 188.9400(s)，

显然自适应算法的要收敛快，精度也高得多。

当采用自适应学习速率进行网络训练时，误差曲线是呈锯齿状下降状，学习速率也是呈锯齿状。这是因为它遵循以下的调整规则：当新误差超过旧误差一定倍数时，学习速率将减少，否则其学习速率保持不变；当新误差小于旧误差时，学习速率将被增加，这样网络总是以最大的可接受的学习速率进行训练。其训练效果比在整个训练过程中固定一个学习速率要好得多。

11.1.3 改进算法的性能比较

保持各参数不变，采用第 2 章中基于非线性优化的训练算法对上述网络进行训练，并与基于标准梯度下降的改进方法，如:附加动量 BP 算法、学习速率可变的 BP 算法和弹性 BP 算法进行性能对比。基于标准数值优化的改进方法有共轭梯度算法，拟牛顿法与 Levenberg_Marquardt 法。作为本节例题中对各种改进算法的性能比较，其训练结果如表 11.4 所示。

表 11.4 不同改进算法的训练结果记录

函数	算法描述	SSE	Epochs	Time(s)
traingda	变学习速率的 BP 算法	0.00099753	6728	178.1800
traingdm	附加动量的 BP 算法	0.00611616	10000	197.2300
traingd	标准梯度下降法(标准 BP 算法)	0.00589787	10000	132.0400
traincgf	Fletcher_powell 共轭梯度法	0.000931746	107	4.5600
traincgp	Polak_Ribiere 共轭梯度法	0.000993415	161	5.9300
traincgb	Powell_Beale 共轭梯度法	0.000964970	129	4.8900
trainscg	Scaled 共轭梯度法,无需进行线性搜索	0.000989956	151	4.5500
trainbfg	BFGs 拟牛顿法	0.000919438	48	2.5200
trainoss	One_Step_Secant 一步割线拟牛顿法	0.000992918	229	6.7000
trainlm	Levenberg_Marquardt 法	0.000931362	28	1.6400
trainrp	弹性 BP 算法	0.000992987	517	7.6900

从表 11.4 中可以看出，附加动量的 BP 算法、变学习速率的 BP 算法以及弹性 BP 算法这三种算法的收敛速度从训练次数上看,后面的算法比前一个算法均高出一个数量级(依次为 10000，6728 和 517)，其中弹性 BP 算法速度快，计算量也不大，非常有效；BFGs 拟牛顿法效果较好，一步割线拟牛顿法与共轭梯度法差不多；在所有的算法中，Levenberg_Marquardt 法的效果最好，它的运算次数及花费时间都是最少的。

人工神经网络训练程序的编写也是网络设计成败的关键步骤，因为它是将设计者的网络设计结构与算法实现相结合确定网络参数的过程。不论哪种算法，基本上都要用到函数的一阶甚至二阶导数，同时涉及到矩阵运算，这些都使得程序编写复杂，运算量巨大。所以在可能的情况下，应当尽量利用通用可靠的标准程序，如国际控制界常用的 MATLAB 环境下神经网络工具箱。本节中给出的所有训练算法及其训练过程，都是在此神经网络工具箱的帮助下完成的。

由 Levenberg_Marquardt 法训练最后获得的网络权值[W1,B1],[W2,B2]的设计结果为：

```
W1=[3.0995 -3.1661;1.2010 -1.1431;1.1149 -1.5871;4.1389 4.1389;
    0.0625 0.3600;0.4979 0.1562;2.4634 2.4634;-1.6657 1.3309;2.7434 3.6439];
B1=[-1.1019;-0.4163;2.0054;0.8540;0.3124;0.8313;0.5292;3.2671;1.5999];
W2=[-1.8156 3.0747 -0.8320 1.0364 3.2925 -0.6920 -0.1764 -0.8911 -1.0187];
B2=[-0.7765];
```

11.2 神经网络在电机非线性补偿中的设计与实现

在运动控制应用中，存在着大量的以电机作为执行机构的机械系统速度和位置的控制。执行机构内部的静摩擦力所产生的非线性，使得速度或力矩输入转向时，导致跟踪误差。如果对此非线性摩擦力没有一个合适的补偿，就可能产生低劣的动态特性。在以微处理器为基础的运动控制系统中，由常规的 PID 控制器所获得的精度可以通过对静摩擦力影响的补偿而进一步得到改善。事实上，神经网络以它的自学习和任意逼近非线性函数的特点，可以用来解决由常规方法很难处理的非线性问题。

本节将神经网络使用在伺服电机的位置控制系统中，所设计的神经网络补偿器对由静摩擦力产生的非线性进行补偿。此神经网络是采用间接设计法完成的。即先设计一个神经网络来作为非线性伺服电机的仿真器，再通过此神经网络来完成对补偿器的训练设计，所以设计过程分两部分：首先训练一个神经网络仿真器，为非线性伺服电机建模，当它满足性能要求时，再将仿真器与另一个神经网络相串联，设计训练神经网络补偿器的权重值，使之成为非线性伺服电机的逆模型。一旦第二个神经网络也满足训练指标，可将此神经网络与常规的 PID 调节器并联使用。常规控制器在负反馈系统中实现期望的动态特性。

在下面几节里，首先描述控制问题。然后，通过所采集的伺服电机的输入输出数据，对非线性伺服电机的神经网络仿真器及补偿器进行训练。所设计的神经网络补偿器已在实际的运动控制的位移控制系统上实现。最后给出神经网络补偿器和常规控制器实验结果之间的有意义的比较。

11.2.1 问题的提出

具有静摩擦力的伺服电机的动力学数学模型可以由下列函数描述：

$$
\begin{aligned}
\frac{\mathrm{d}\theta}{\mathrm{d}t} &= \omega \\
\frac{\mathrm{d}\omega}{\mathrm{d}t} &= f_1(\theta,\omega,T) + f_2(\theta,\omega,T_f)
\end{aligned}
\tag{11.1}
$$

其中，θ 和 ω 分别代表角位移和角速度；T 和 T_f 分别表示与控制电流成正比的电机力矩和力矩干扰；f_1 是一个以电机参数为变量的线性函数，而 f_2 是一个受静摩擦力影响的非线性函数。图 11.1 所示为正弦输入信号下，伺服电机受摩擦力影响的输出位移信号。

问题是：设计一个神经网络补偿器，使其产生控制力矩 $T(k)$ 来获得期望的动、静态特性。这个神经网络补偿器最终通过与一个常规的 PID 控制器相并联来实现整个控制系统，神经网络在其中起伺服电机逆模型的作用。

11.2.2 伺服电机神经网络仿真器的设计

伺服电机的模型可以通过训练一个采用具有图 11.1 所示的输入输出变量的前向多层

神经网络来实现，此神经网络的训练过程如图 11.2 所示。目标是获得实际伺服电机位移 $\theta(k)$ 的满意近似值。可采用标准的反向传播法来训练神经网络模型 $\hat{P}$，神经网络的权重值 W 通过使得下列误差函数值为最小来进行调整：

$$J = E[\theta(k)-\hat{\theta}(k)]^2 = E[\theta(k)-f(W * X)]^2 \tag{10.2}$$

此处，$\hat{\theta}(k)$ 是训练的输出量，$X(j) = [\,x(1) \ldots x(n)]^T$ 是神经元输入矢量，而 $W(i,j) = [\,w(i,1)\ w(i,2) \ldots w(i,n)\,]$ 是在 k-1 次训练时的权重系数，n 是输入层的输入变量的数目，$i = 1,2,\ldots,s$ 是隐含层中的神经元的数目。本例采用一个两层神经网络，在隐含层使用 S 型函数，在输出层使用的是线性函数，隐含层中采用 5 个神经元，输出层中有 1 个神经元，如图 11.2 所示。

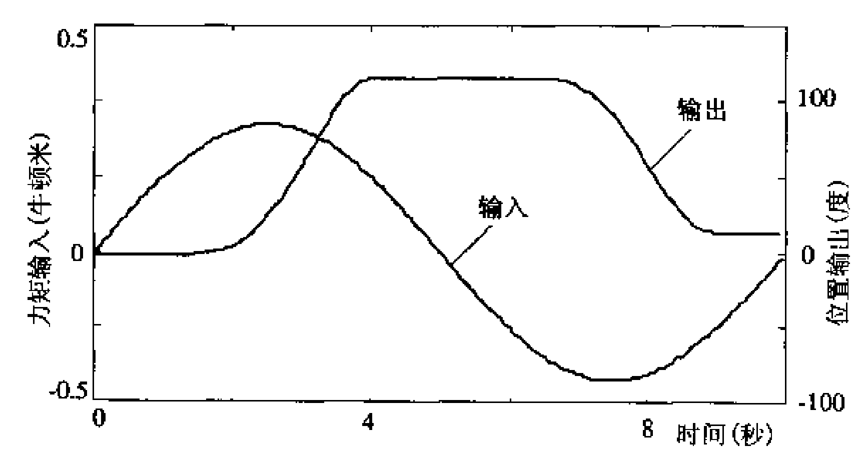


图 11.1 被控伺服电机的输入输出特性

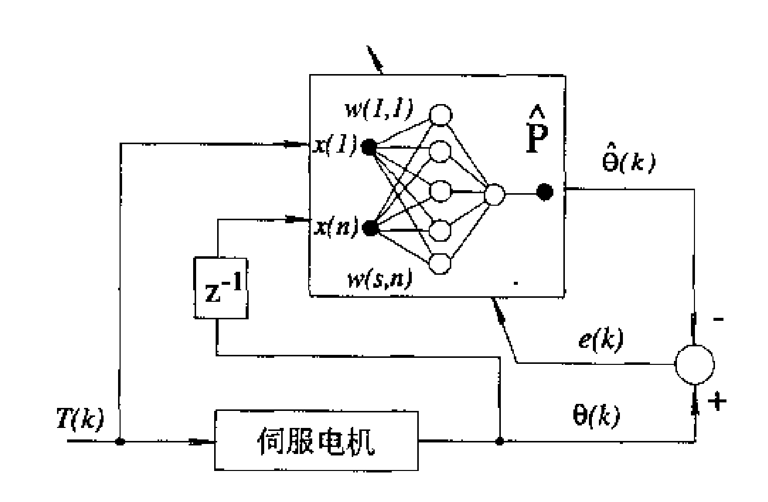
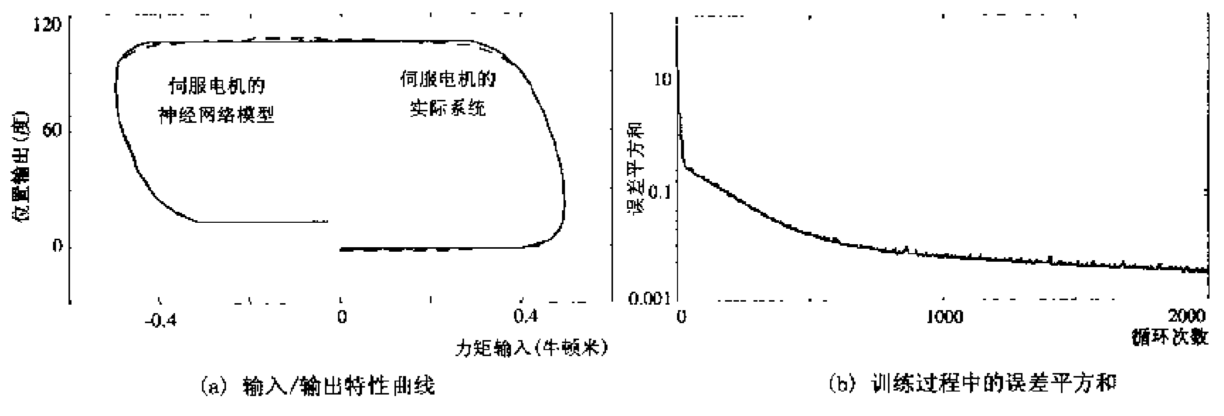


图 11.2 伺服电机模型的训练过程

网络的训练在误差平方和小于 0.001 或当循环训练次数达到所设最大值 2000 次时终止，网络的训练过程通过 MATLAB 神经网络工具箱来完成。图 11.3(a)所示伺服电机与神经网络仿真器的输入/输出特性曲线；图 11.3(b)为神经网络训练过程中的误差平方和的记录，经过训练，神经网络可以用来对不同的输入信号输出近似的伺服电机的行为。



(a) 输入/输出特性曲线　　　(b) 训练过程中的误差平方和

图 11.3 伺服电机神经网络仿真器的训练记录

11.2.3 神经网络补偿器的设计

在前一节中，获得了伺服电机的神经网络仿真器，本节通过将仿真器与另一个神经网络串联来训练伺服电机的补偿器。它是采用伺服电机的期望输出与伺服电机训练输出之间的差来调整权值的，所以训练过程中使用仿真器来代替伺服电机，并且用仿真器输出的一阶延迟作为神经网络补偿器的另一个输入，训练过程中，仿真器的权值保持不变。图 11.4 为神经网络补偿器的训练过程。

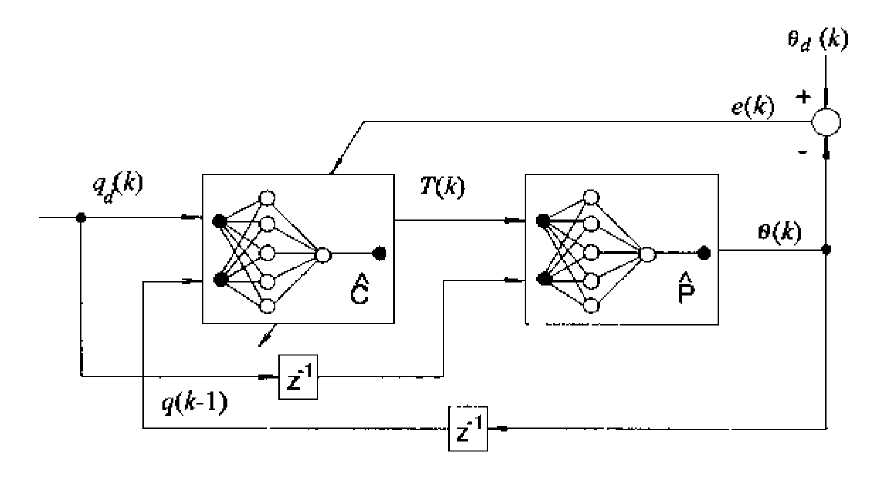


图 11.4 神经网络补偿器的训练过程

开始时，神经网络的输入是期望角位移 $\theta_d(1)$ 和来自仿真器的初始输出 $\theta(0)$。因为神经网络没有经过训练，它给出不正确的输出控制力矩 $T(1)$，所以仿真器输出 $\theta(1)$ 也是不正确的。此时，$\theta_d(1)$-$\theta(1)$ 之间的误差通过反向传播过程，被用来调整神经网络补偿器的权重值。然后输出 $\theta(1)$ 用来做为神经网络补偿器的新输入变量，它产生一个新控制力矩 $T(2)$，并且从仿真器得到新值，再次修正神经网络补偿器的权值，这个过程一直重复到获得满意的训练结果。图 11.5 是期望位移和从神经网络补偿器与仿真器串联后输出值的比较。

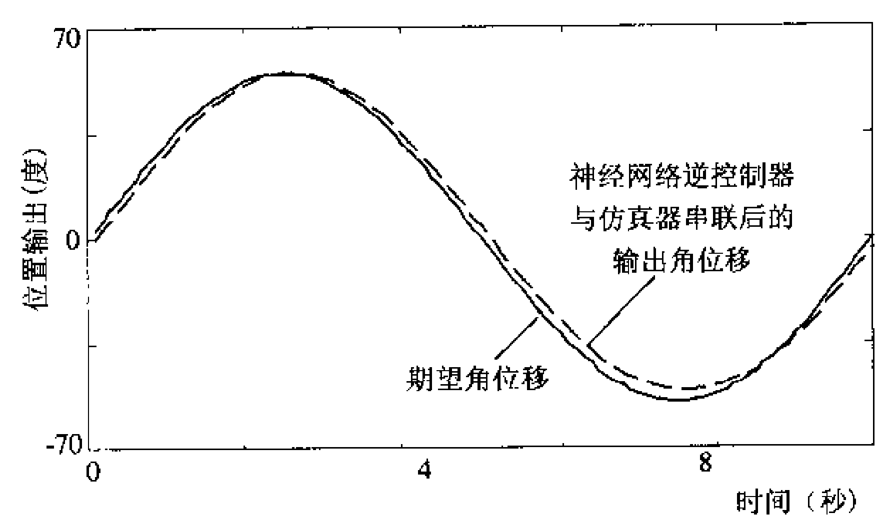


图 11.5 期望位移和从神经网络补偿器与仿真器串联后输出值的比较

11.2.4 神经网络控制系统

如前所述，在一般的伺服电机模型中，含有一个线性部分和一个非线性部分，第一部分代表忽略了所有的非线性时的线性电机械电路特性，第二部分描述在电机转子角位移中的非线性的影响，如果线性模型可以用传递函数 $P^*(s)$ 来表示，而对非线性影响的补偿可以通过将干扰力矩 T_f 加在线性模型的输出力矩上来实现，干扰力矩部分有关静摩擦力可以用 T^*_f来表示，而用 T_l 来表示由其他不同的源所引起的干扰，有：

$$T_f = T^*_f + T_l \tag{11.3}$$

传递函数 $P^*(s)$ 有下列结构：

$$P^*(s) = \frac{K}{s(Js+B)} \tag{11.4}$$

其中，J 和 B 分别表示惯性和粘摩擦系数，K 为放大系数，它们的值可以通过施加合适的控制变量，采用标准的 ARMAX 辨识过程求得。

一旦具有逆模型的神经网络被正确地训练出，它就可以给出力矩 T ，其中包括控制命令和摩擦力矩 T^*_f 。在其他干扰力矩被忽略的情况下，这个神经网络补偿器与伺服电机的串联所实现的被控装置，具有等于 1 的输入/输出特性。为了获得特性指标，只有由其他源所引起的干扰力矩 T_l 的影响需要被消除，为了达到此目的，可以增加一个负反馈回路并采用一个适当的调节器 $G(S)$。该调节器的传递函数的设计可以根据性能指标，采用常规的或先进控制策略来实现，最简单的方式是应用一个 PID 控制器，并根据伺服电机的参数值来对其参数进行调整，图 11.6 所示为被控伺服电机的负反馈控制系统方框图，其中神经网络与负反馈控制器相并联。因此，通过神经网络的实施，补偿器主要用来消除非线性摩擦力的影响，而闭环动力学性能是通过负反馈控制器来达到的。

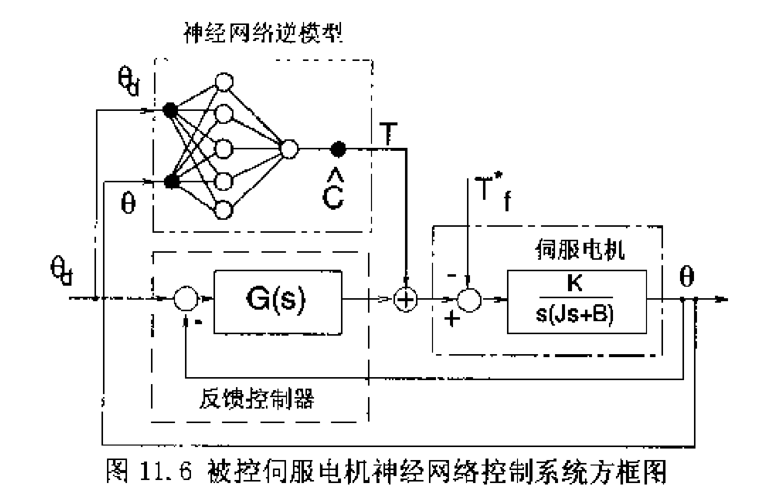
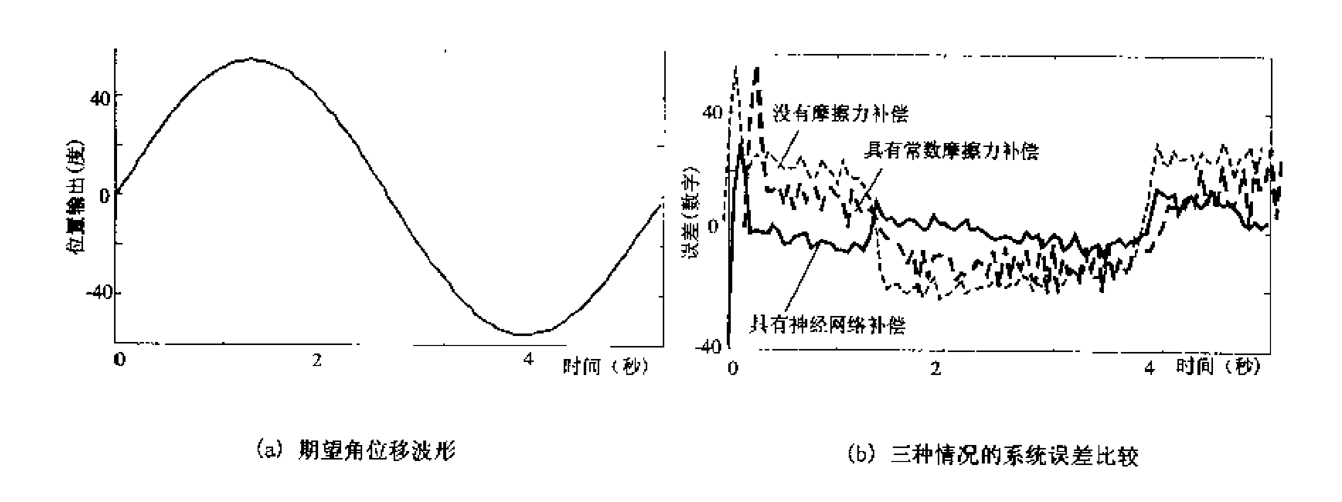


图 11.6 被控伺服电机神经网络控制系统方框图

11.2.5 实验测试结果

实验是基于 IBM PC-386，内插有 DSP-32C 板。直流伺服电机是由 PWM 功率放大器提供的电流型控制，伺服电机的角位移范围为 0 到 360 度，目标当静误差|□-□|小于 0.5 度时即可达到。控制策略已由 DSP32C 汇编语言编成程序，所应用的神经网络补偿器由 DSP 来实现；所有的变量都经过 D/A 转换，放大后产生力矩控制信号；位移是由位测传感器测量后送入一个 16 位 A/D 变换器，实验的实施还用到 MATLAB 和 C 语言。

实验结果如图 11.7 所示，图 11.7(a)为期望的角位移波形。为了进行比较，我们还做了另外两个实验：仅使用 PID 控制器而没有采用摩擦力补偿和采用常数 T^*_f 对摩擦力进行补偿。在图 11.7(b)中，对此三种控制情况的系统误差进行了比较，其中，采用数字信



(a) 期望角位移波形　　(b) 三种情况的系统误差比较

图 11.7 实验结果和比较

号处理器的数字单位来表示它们，从中可以很容易地看出，当没有摩擦力补偿时，系统误差范围在 ±200 数字；而当使用常数 T^*_f 时，系统误差范围为 ±100 数字；当采用神经网络进行补偿时，实际的系统误差接近于 0。这证实，采用神经网络进行补偿可以得到更好的结果。

11.2.6 小结

静摩擦力是在小体积和低价伺服电机中不可避免的现象，在运动控制中使用伺服电机，如果采用常规的控制策略，动态性能则会下降。改善性能指标的唯一方法是应用先进控制策略来达到对静态摩擦力进行补偿的目的，可以采用数字控制器的采用来实施各种不同的可行方法。从终端用户的观点來看，主要的问题是在保持性能指标质量的前提下，提出一个对伺服电机具有调节性的控制策略；从生产者的观点来看，主要问题是通过合适的价格和所要达到的指标来选择控制器的不同的实施方法；从控制策略设计者的角度来看，主要问题是获得不同技术的可能性。对此类控制器的设计，需要较深的控制理论以及系统应用的能力。传统的办法可以应用在常规控制器中，但不可能获得相同的性能特性指标。

本节中所给出的控制系统是对静摩擦力进行补偿的一种可行的实施，用训练的神经网络补偿器来消除非线性静摩擦力的影响，与神经网络相并联的负反馈控制器可以用来获得期望的动态特性。实验结果表明，相对于常规的补偿器，此控制策略能够使系统达到更高的性能指标。

第 12 章 模糊神经网络

12.1 引 言

模糊系统已经广泛地应用在模糊控制器、模式识别、模糊辨识和信号处理中。模糊逻辑方法的优点在于它的逻辑性和透明性上，使人们很容易将先前已知的有关的系统知识结合到模糊规则中。但是，模糊系统必须包括人为选定的模糊隶属函数的模糊化过程和采用适当方法的模糊规则的推理过程，每一过程对系统的执行性能都有影响，而且模糊系统中模糊规则的前提和结论部分通常都是模糊子集，解模糊也是一项复杂的工作，一方面，模糊系统输入/输出关系式是高度非线性的，要想得到一个满意的输入/输出关系式，在众多需要调节的参数面前，再有经验的专家也难以胜任。而另一方面，人工神经网络的最大益处就在于它善于对网络参数的自适应学习，并且具有并行处理及泛化能力。这很自然促使研究者进行将模糊逻辑和神经网络结合成一个系统的研究，并称其为模糊神经网络或模糊神经系统。通过神经网络实现的模糊逻辑系统结构具有模糊逻辑推理功能，同时网络的权值也具有明确的模糊逻辑意义，从而达到以神经网络及模糊逻辑各自的优点弥补对方不足的目的。

随着维数增加，模糊系统面临规则数目的指数增长。过去的大多数模糊应用是低维数，一般所使用的仅仅是 2 到 3 个输入和 1 个输出，而所使用的模糊标记大多是 5 或 7，因此，使用的规则不多于 150 条。但当要处理较复杂的金融、医药或机器制造系统时，简单模糊系统可能运转不灵。这样，模糊系统专家们不仅需要大量的规则，而且需要大量好的规则。

学习可能有助于在有限条规则下就能解决这样的难题，学习可能移动规则从而调整模糊系统，学习可能来自于一个数学算法，或一个搜索软件，或一个专家的猜测，或训练和误差。学习是获得新的更好的规则的一个手段。学习改变了如果部分集的形状或位置，或者改变则部分集的形状或位置，或两者都有所改变。新集给出新规则，新规则给出新系统。

本章从模糊关系式入手，分别对直接的模糊神经网络的实现、采用 sugeno 模糊推理法实现的模糊系统的特点、B 样条模糊神经网络以及径向基函数神经网络等系统之间的相互联系，以及各自所表现出来的优缺点进行了性能分析与对比研究，以此达到使人们对模糊神经网络有更加深入的认识，更好地设计网络、利用网络系统完成不同应用的目的。

12.2 模糊系统的关系式

考虑一个一般形式的模糊系统，其输入是 $x \in R^n$，输出为 $y(x) \in R$。x_i 和 y 的论域

分别是 $x_i \in R$ $(i=1,2,\cdots,n)$ 和 $y\in R$。x_i 的语言值是 $A_i^{k_i}$（$k_i=1,2,\cdots,m_i$ 和 $i=1,2,\cdots,n$），模糊规则的数量为 $p=m_1m_2\cdots m_n$。第 k 条规则为：

$$\begin{aligned}
&\text{如果 } x_1 \text{ 是 } A_1^{k_1}(x_1)\text{，并且}\\
&\quad x_2 \text{ 是 } A_2^{k_2}(x_2)\text{，并且}\\
&\quad \cdots\\
&\quad x_n \text{ 是 } A_n^{k_n}(x_n)\text{，}\\
&\text{那么 } y_k \text{ 为 } y_k(x)
\end{aligned} \tag{12.1}$$

其中，$y_k(x)\in Y$，k 为规则基的数目，有 $k=1,2,\cdots,p$，每个 k 对应一个有阶序列 $k_1,\cdots,k_i,\cdots,k_n$，其中 $k_i=1,2,\cdots,m_i$。若采用乘积法进行与操作，那么第 k 条模糊规则前提部分的真值 $\mu_k(x)$ 可以表示为：

$$\mu_k(x)=A_1^{k_1}(x_1)\cdot A_2^{k_2}(x_2)\cdot\cdots\cdot A_n^{k_n}(x_n)=\prod_{i=1}^{n}A_i^{k_i}(x_i) \tag{12.2}$$

从前提部分到结论部分的推理可产生一个结论 C，它是一个离散的、具有有限点数的模糊子集：

$$C=\{u_k/y_k \mid k=1,2,\cdots,p\}$$

其中，u_k/y_k 表示在点 $y_k\in Y$ 上的 C 的隶属度是 u_k。

采用质心法对模糊集 C 进行解模糊，系统的实际输出为：

$$y(x)=\frac{\sum\limits_{k=1}^{p}u_k(x)y_k(x)}{\sum\limits_{k=1}^{p}u_k(x)} \tag{12.3}$$

将（12.2）式中 $\mu_k(x)$ 的值代入上式分母，上式可重写为：

$$\begin{aligned}
\sum_{k=1}^{p}u_k(x)&=\sum_{k_1=1}^{m_1}\sum_{k_2=1}^{m_2}\cdots\sum_{k_n=1}^{m_n}A_1^{k_1}(x_1)A_2^{k_2}(x_2)\cdots A_n^{k_n}(x_n)\\
&=\sum_{k_1=1}^{m_1}A_1^{k_1}(x_1)\sum_{k_2=2}^{m_2}A_2^{k_2}(x_2)\cdots\sum_{k_n=1}^{m_n}A_n^{k_n}(x_n)
\end{aligned} \tag{12.4}$$

所以（12.3）式变为：

$$y(x)=\frac{\sum\limits_{k=1}^{p}\prod\limits_{i=1}^{n}A_i^{k_i}(x_i)y_k(x)}{\sum\limits_{k_1=1}^{m_1}A_1^{k_1}(x_1)\sum\limits_{k_2=2}^{m_1}A_2^{k_2}(x_2)\cdots\sum\limits_{k_n=1}^{m_n}A_n^{k_n}(x_n)} \tag{12.5}$$

由此可见，模糊系统的输入/输出的求解关系式是相当复杂和费时的。另一种人们常用的模糊推理方法是所谓的最小推理，即将关系式（12.2）中 $\mu_k(x)$ 的操作变为取小运算，然后再通过（12.3）式进行解模糊，并通过制表法来设计模糊逻辑表。不过这种取小运算

将有用信息丢失，利用信息率较低。而采用（12.2）式中的乘积操作考虑了所有信息的影响，同时便于用神经网络来实现模糊系统，因而得到广泛的应用。

12.3 用神经网络直接实现的模糊系统

模糊集合理论提供了系统的、以语言表示这类信息的计算工具，通过使用由隶属函数表示的语言变量，可以进行数值计算。不过在模糊推理系统中,合理选择 if-then 规则是模糊推理系统的关键因素。尽管模糊推理系统拥有模糊 if-then 格式的结构化知识表示，但是它们缺少对变化的外部环境进行适应的能力。因此，人们将神经元网络中学习的概念引入模糊推理系统，从而产生神经模糊系统，它是神经模糊控制系统中的关键技术。

为了能够用计算机对模糊逻辑进行实时操作，一个很自然的想法是通过神经网络来实现模糊系统。在采用神经网络实现模糊化、模糊推理和解模糊化过程中，为了运算简单，最直接的模糊神经网络系统中采用的模糊推理方式是用乘积—求和代替模糊系统中常用的最小—最大推理法，且输出变量的隶属函数取单值型。

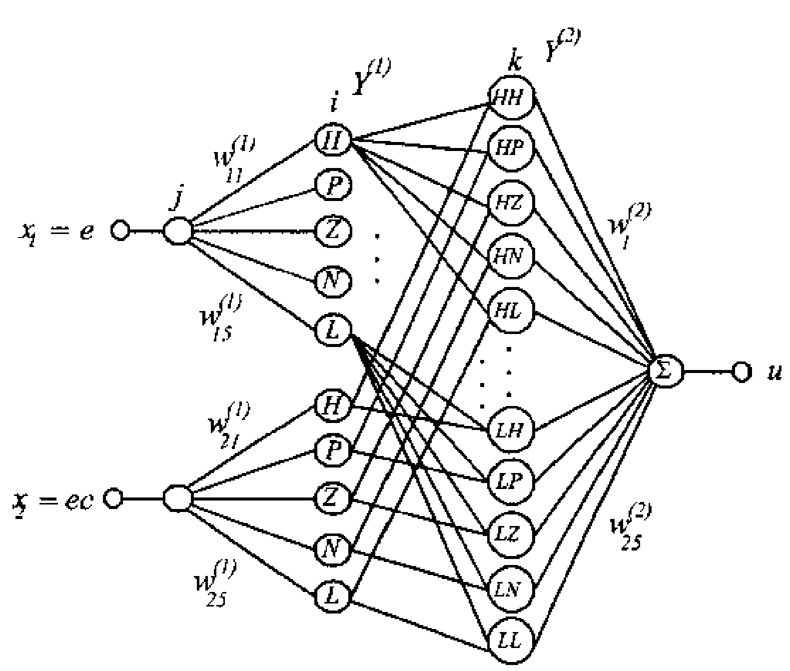


图 12.1 具有模糊逻辑功能的神经网络的结构

假定输入为 x_1 和 x_2，模糊标记取高（H）、正（P）、零(Z)、负（N）和低(L)五个，由此可构造一个具有模糊功能的神经网络，如图 12.1 所示。

由图 12.1 可以看出，它是一个具有输入层、中间层和输出层的网络结构，而输入 x_1 和 x_2 各自被送入一个前向人工神经网络，该网络的激活函数的形状即代表模糊隶属函数的形状，可由设计者确定采用线性或指数型，分别可对应模糊隶属函数的三角型和高斯型；输入层神经元的个数代表模糊标记数，也是由设计者选定。通过调节输入层网络的权值 $W^{(1)}$和偏差 B，可以达到调节激活函数的宽度及其中心位置的目的。

由输入层输出的 H、P、Z、N 和 L 分别代表模糊化后的结果——隶属度。中间层执

行的是将模糊化得到的隶属度两两相乘的功能。由 x_1 得到的 H 值与 x_2 得到的 H 值相乘获得 HH 值，由 x_1 得到的 H 值与 x_2 得到的 P 值相乘获得 HP 值，以此类推，可获得由 x_1 得到的 L 值与 x_2 得到的 L 值相乘得到 LL 值。所以在图 12.1 中所看到的 HH、HP、…、HL 等节点是简单的乘法单元，功能是计算出相应输入的乘积。最后乘法单元的输出通过 $w_1^{(2)}$ 到 $w_9^{(2)}$的权值送到一个求和节点中，于是整个网络的输出值是：

$$S = H \cdot H \cdot w_1^{(2)} + H \cdot P \cdot w_2^{(2)} + \cdots + L \cdot N \cdot w_8^{(2)} + L \cdot L \cdot w_9^{(2)} \tag{12.6}$$

这就实现了乘积—求和的模糊推理过程。

此网络的最大优点是用神经网络实现了模糊系统，从而可以通过训练来求得其中的参数，不必依赖专家经验人工事先确定，做到了自动确定系统的参数。系统的隶属函数的中心、宽度以及规则等全部参数同时得以自寻优。

该模糊神经网络的实现是通过对网络采用监督式训练来确定输入层的权值、偏差以及输出层的权值。由于该网络不是标准的全联接，加上模糊标记一般情况下比较多，所以网络的训练速度通常是相当慢的，对于某些复杂的输入/输出关系，网络的训练常常很难收敛到期望的极小值。另外，由于网络中代表模糊标记的神经元数必须由设计者事先确定，它的数目对网络的精度也是有影响的。

12.4 Sugeno 模糊推理法

迄今为止，人们常用的模糊推理过程被称为玛达尼模糊推理法，玛达尼推理中的结论部分是模糊集合，对模糊集合进行处理后，每个输出变量需要被解模糊。在实践中人们认识到采用单值型隶属函数可以简化计算，提高解模糊的有效性。Sugeno（或 Takagi-Sugeno-Kang）推理法对结论就是采用单值型隶属函数的。Sugeno 法的前提部分与玛达尼推理法有相同的结构，而结论部分不同于原来的模糊集合，它用一个前提部分变量的多项式表示，是前提变量的线性函数，并采用加权平均法求解模糊。

零阶 Sugeno 模糊系统的典型模糊规则有如下形式：

$$\text{如果}x_1\text{为}A\text{，并且}x_2\text{为}B\text{，那么}y = k \tag{12.7}$$

此处 A 和 B 为前提中的模糊集合，而 k 为结论中一个可调参数。一阶 Sugeno 模糊系统具有的模糊规则形式为：

$$\text{如果}x_1\text{为}A\text{，并且}x_2\text{为}B\text{，那么}y = px_1 + qx_2 + r \tag{12.8}$$

此处 A 和 B 为前提中的模糊集合，而 p、q 和 r 为可调参数。想象此一阶系统运作方式的简单方法就是把每一条规则认为是定义一个“移动单值”的位置，即取决于输入值的单值输出可以在输出空间中移动。

作为对比，这里需要指出的是：如果将模糊规则（12.1）中的 $y_k(x)$设置成一个由输入矢量表示的线性组合，即：

$$y_k(x)=a_1^k x_1 + a_2^k x_2 + \cdots a_n^k x_n \tag{12.9}$$

那么由（13.3）式所示的则为采用 Sugeno 推理法时的系统输出。

当每条规则在系统输入空间中是线性独立时，Sugeno 法可以作为对一个非线性动力学系统进行不同操作条件的复合线性控制器。另外，一个 Sugeno 系统也适用于非线性系统的建模，它是通过插入复合线性模型来实现的。

由于输出的线性组合，使得由 Sugeno 推理法求得的模糊系统比用玛达尼法具有更好的效果。实际上仅从模糊系统角度来看，Sugeno 推理法同样存在如何确定有效的模糊规则的问题。但当把神经网络与其相结合起来后，Sugeno 推理法才显示出它的优越性。

表 12.1 中给出 Sugeno 推理法与玛达尼推理法性能的对比结论。有关采用 Sugeno 推理法进行网络设计与训练的例子，请看第 13 章。

表 12.1 Sugeno 与玛达尼推理法性能对比

	Sugeno 推理法	玛达尼推理法
优点	①有效的计算（直接由输入到输出的点到点的计算） ②与线性技术（如 PID 等）最优化技术及自适应技术能一起很好的工作（因其本身就是线性复合技术） ③保证输出空间的连续性 ④更适合进行数学分析	①较直观 ②被初学者广泛的应用 ③较适合于人工输入规则的情况
缺点	①参数的确定仍然很麻烦	①运算复杂 ②丢失信息较多

12.5 B 样条模糊神经网络

通过分析模糊系统的推理过程可知，系统主要有三个可进行选择的部分：隶属函数的形状、模糊规则和解模糊过程。用神经网络实现模糊系统的另一思想是对模糊系统中某一部分的参数实现自动调节。

对于隶属函数的调整，B 样条基函数不失为一种较好的选择，因为它具有期望的数学特性，如局部紧支撑、数学计算的简单和单位分割性。将 B 样条基函数作为隶属函数，构造一个 B 样条模糊神经网络系统也是人们一直关注的。

12.5.1 B 样条函数及其网络

B 样条是基本样条（Basic Spline）的简称。B 样条算法最重要的特性是由其函数的形状而能够产生光滑的输出。另外，B 样条函数是所有样条函数中具有最小局部支撑的样条函数，所以 B 样条基函数可以精确多项式分段插值的方式，对给定的输入/输出数据进行光滑的曲线拟合。1972 年德布尔（de Boor）和考克斯（Cox）分别独立地给出关于 B 样条计算的标准方法，证明 B 样条网络可以任意精度逼近一个连续的函数。

给定一组单变量 x 的节点序列：$x_1 < x_2 < \cdots < x_{N+m}$，那么在区间 $[x_m, x_{N+1}]$ 之间就可以唯一确定 $N+m$ 个 m 阶线性不相关的 B 样条基函数，一个连续函数 $y(x)$ 就可以用 $l = N+m$ 个 B 样条基函数的线性组合来近似的表示为：

$$y(x) \approx \sum_{i=1}^{l} \omega_i B_{i,m}(x), \qquad x \in [x_m, x_{N+1}] \tag{12.10}$$

其中，ω_i 为第 i 个 k 阶 B 样条基函数 $B_{i,k}(x)$ 的权值；N 为拟合点的顶点数。当具体计算其函数值时，这些权值必须被估值。

B 样条基函数满足以下迭代关系式：

$$\begin{cases} B_{i,1}(x) = \begin{cases} 1, & x \in (x_i, x_{i+1}] \\ 0, & x \notin (x_i, x_{i+1}] \end{cases} \\ B_{i,k}(x) = \dfrac{x - x_i}{x_{i+k-1} - x_i} B_{i,k-1}(x) + \dfrac{x_{i+k} - x}{x_{i+k} - x_{i+1}} B_{i+1,k-1}(x), \quad k = 2, \cdots m+1 \end{cases} \tag{12.11}$$

由（12.11）式可以看出，一个 k 阶 B 样条函数是通过比它低一阶次的 $k-1$ 次多项式曲线拟合获得的。

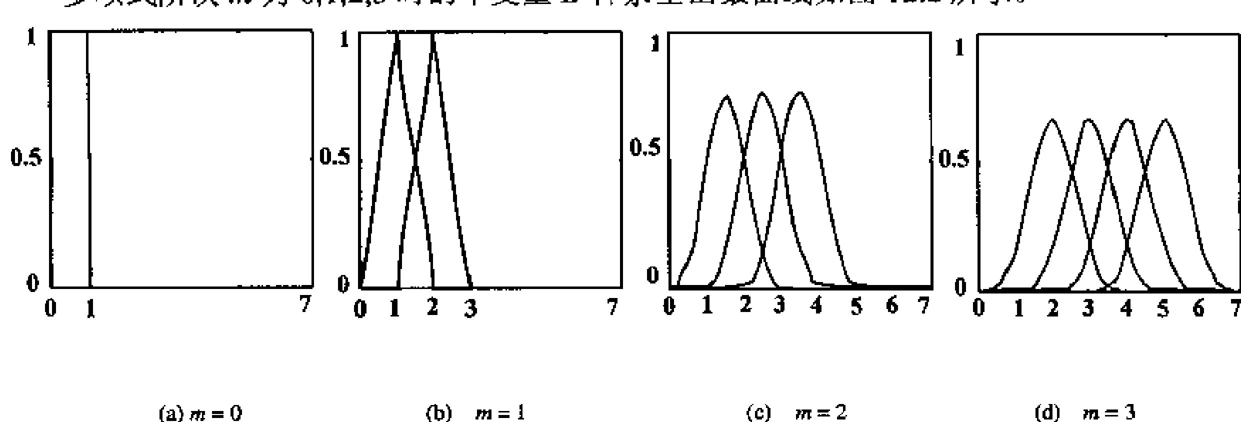
B 样条函数是非负的，且局部支撑的，即：

$$B_{i,k}(x) \begin{cases} > 0, & x \in [x_i, x_{i+m}) \\ = 0, & 其他 \end{cases} \tag{12.12}$$

且它们形成一个单份分割，即：

$$\sum_{i=1}^{N} B_{i,m}(x) \equiv 1, \qquad x \in [x_m, x_{N+1}) \tag{12.13}$$

多项式阶次 m 为 0,1,2,3 时的单变量 B 样条基函数曲线如图 12.2 所示。



(a) $m=0$　(b) $m=1$　(c) $m=2$　(d) $m=3$

图 12.2　阶次 m 为 0, 1, 2, 3 时的 B 样条基函数曲线图

由此可见，B 样条的阶次总是比用来表示此阶 B 样条的多项式曲线的阶次高一阶，如一阶 B 样条是由 $m=0$ 阶直线表示的；二阶 B 样条是由 $m=1$ 阶斜线表示的。

对于多变量 B 样条，可以假定 n 维输入矢量 $x = [x_1, x_2, \cdots, x_n]^t$，在每个输入轴上定义

单变量 B 样条基函数 $B_{i,m_i}^{k_i}(x_i), k_i=1,2,\cdots,m_i, i=1,2,\cdots,n$，那么，第 k 个多变量 B 样条基函数 $M_k(x)$ 由 n 个单变量基函数 $B_{i,m_i}^{k_i}(x_i)$ 乘积（张积）组成：

$$M_k(x)=B_{1,m_1}^{k_1}(x_1)B_{2,m_2}^{k_2}(x_2)\cdots B_{n,m_n}^{k_n}(x_n)=\prod_{i=1}^{n} B_{i,m_i}^{k_i}(x_i) \tag{12.14}$$

$M_k(x)$的总数为 $p=m_1m_2\ldots m_n$，所以 $k=1,2,\cdots,p$。

张积 B 样条基函数仍然保留（12.12）式和（12.13）式的特性。

那么一个多变量函数 $y(x)$可以用多变量 B 样条基函数的线性组合来逼近：

$$y(x)\approx\sum_{k=1}^{p} M_k(x)w_k=\sum_{k=1}^{p}\prod_{i=1}^{n} B_{i,m_i}^{k_i}(x_i)w_k \tag{12.15}$$

仔细观察可以发现，B 样条基函数的阶数 k 与不同形状的模糊隶属函数相对应，如当 $k=2$时，用 B 样条作的曲线正好代表三角形隶属函数，当 $k=3$ 时，所作曲线与二次型隶属函数相似。另一方面，所取的顶点数 N 对应于模糊标记数。节点数则在一定范围内决定了个隶属函数的宽度。不过，高于一阶的 B 样条函数并不一定是一个正则（normal）模糊集，即 B 样条函数的最大值达不到 1，而通常模糊系统理论要求隶属函数为正则的。为了满足这一要求，可对 B 样条基函数乘上一个正则化因子 λ，以使其最大值为 1，并取隶属函数为：

$$\left.\begin{aligned}&\lambda=1\Big/\sup_{x\in x_i} B_{i,m_i}^{k_i}(x_i)\\&A_i^{k_i}(x_i)=\lambda B_{i,m_i}^{k_i}(x_i), \qquad i=1,2,\cdots,n\end{aligned}\right\} \tag{12.16}$$

将（12. 16）式代入模糊系统输入/输出关系（12. 5）式中：

$$\begin{aligned}y(x)&=\frac{\sum_{k=1}^{p}\prod_{i=1}^{n}\lambda B_{i,m_i}^{k_i}(x_i)y_k(x)}{\sum_{k_1=1}^{m_1}\lambda B_{1,m_1}^{k_1}(x_1)\sum_{k_2=1}^{m_2}\lambda B_{2,m_2}^{k_2}(x_2)\cdots\sum_{k_n=1}^{m_n}\lambda B_{n,m_n}^{k_n}(x_n)}\\&=\frac{\lambda^n\sum_{k=1}^{p}\prod_{i=1}^{n} B_{i,m_i}^{k_i}(x_i)y_k(x)}{\lambda^n\sum_{k_1=1}^{m_1} B_{1,m_1}^{k_1}(x_1)\sum_{k_2=1}^{m_2} B_{2,m_2}^{k_2}(x_2)\cdots\sum_{k_n=1}^{m_n} B_{n,m_n}^{k_n}(x_n)}\\&=\sum_{k=1}^{p}\prod_{i=1}^{n} B_{i,m_i}^{k_i}(x_i)y_k(x)\end{aligned} \tag{12.17}$$

与(12.15)式完全相同。由（12.17）式可以看出：采用 B 样条函数作为模糊系统的隶属函数所得到的输入/输出关系式与正则化因子无关，与用多变量 B 样条逼近多变量函数 $y(x)$ 的表达式完全相同。换句话说，用 B 样条逼近函数的表达式所代表的就是一个模糊系统。两者研究的出发点虽然不同，但达到相同的结果，解决相同的问题。

由（12.17）式还可看出，我们可以通过设计求解一个 B 样条网络函数来建立一个模糊系统。因为两者所表达的实质内容是完全等价的。设计者只要事先确定了多项式的阶数 m（即隶属函数的形状），拟合点的顶点数（即模糊标记数），被拟合的点至少必须有 $N+m$ 个。网络函数中唯一需要优化的是权值系数 w_k，它代表模糊系统中推理规则的结论部分，它的选取可以通过不同的优化算法来求解，以达到优化模糊规则的目的。由于 B 样条网络采用多项式计算，使其比单纯的用神经网络的指数函数作为隶属函数的模糊系统的求解节省时间。

由以上分析可得出如下结论：

单变量 B 样条基函数精确地再现了模糊逻辑控制所使用的不同形状的隶属函数,即通过选取不同的阶次,可以用 B 样条基函数的多项式递推公式来计算图 12.1 模糊神经网络中输入层的输出值：H、P、Z、N、L 等。再者，比较(12.6)式和(12.17)式，可以发现,一个多变量 B 样条函数正是一个由乘积求和推理方式组成的模糊神经网络的输入/输出关系的算式。即多变量 B 样条网络在结构上与模糊神经网络之间存在着相等的对应关系。 在多变量 B 样条网络算式(12.17)中，$B_{j,k_j}(x_j)$ 对应于单变量的隶属函数值，$\prod_{j=1}^{n} B_{j,k_j}(x_j)\cdot w_k$ 对应乘积规则，网络通过最后的求和进行了中心法逆模糊的过程。所以 B 样条网络结构与模糊神经网络结构在处理信息的能力上是等价的。B 样条网络的这种透明度是一般神经网络所不具备的，也是一个极有价值的特性，因为它允许设计者用自然的、一般由专家解释其行为时所使用的模糊术语来初始化网络，并通过优化算法对参数进行优化，且仅采用多项式计算，省去了一般神经网络中的非线性指数激活转移函数的复杂运行，从而简单易行和省时。采用多变量 B 样条网络建立的模糊系统，相当于采用 B 样条基函数作为隶属函数，采用优化算法对模糊规则进行自动寻优的模糊系统。优点是：算法相对省时，但缺点是：①隶属函数的模糊标记数事先必须确定；②各隶属函数之间是等距离的，而不是自动寻优的。

12.5.2 B 样条模糊神经网络控制器的设计

在 B 样条网络的设计中，设计者必须确定两个参数：组成 B 样条基函数的多项式的阶数 m 以及单变量 B 样条函数的顶点个数 N，由此可求得单变量 B 样条函数所需的节点数 $l = N + m$。对应于模糊神经网络，阶数 m 表示不同形状的隶属函数，当 $m = 1$ 时，代表三角形隶属函数；当 m 取 2 时，代表二次型隶属函数；而顶点数 N 对应于模糊标记数。节点数则在一定范围内决定了各隶属函数的宽度（每一个 B 样条基函数落在 x 轴上所占节点数为 $m+1$）。

对于模糊神经网络控制器的设计，其输入变量可选为误差 e 和误差的变化 ec。若取阶数 $m_e = m_{ec} = 3$，顶点数 $N_e = N_{ec} = 5$，则节点数 $l_e = l_{ec} = 3 + 5 = 8$，在[-1,1]内等距离取值，由此可得网络控制器的控制量 $u(e,ec)$ 的计算公式为：

$$u(e,ec)=\sum_{k=1}^{N_e \cdot N_{ec}} N_k(e,ec)\cdot w_k=\sum_{k=1}^{N_e \cdot N_{ec}} B_{e,k_e}(e)B_{ec,k_{ec}}(ec)w_k \tag{12.18}$$

其中，$k_e=1, 2, \ldots, m_e$，$k_{ec}=1, 2, \ldots, m_{ec}$。网络结构如图 12.3 所示。

通过训练或优化算法可以确定(12.18)式中的权值参数 w_k。然而由于用于训练的数据 u 实际上是无法知道的，为此，可采用于闭环的间接方式，将所设计的控制器与过程相串联，组成闭环负反馈回路，并以系统参考输入作为样本输入数据，而将系统输出作为实际输出，以两者之间的误差平方和为最小作为训练的目标，并采用遗传算法优化 B 样条模糊神经网络控制器的权值 w_k。因为需要用到被控过程，所以被控过程模型的辨识也是控制器设计的一部分，并在优化控制权值 w_k 时，假定所辨识的过程参数是正确的，且保持不变。在我们的实际应用中，B 样条模糊神经网络控制器的设计过程是在 MATLAB 环境下进行的。具体设计过程如下：

①首先获得实际被控过程的一组输入/输出数据，然后利用神经网络工具包对被控过程的输入/输出进行辨识；

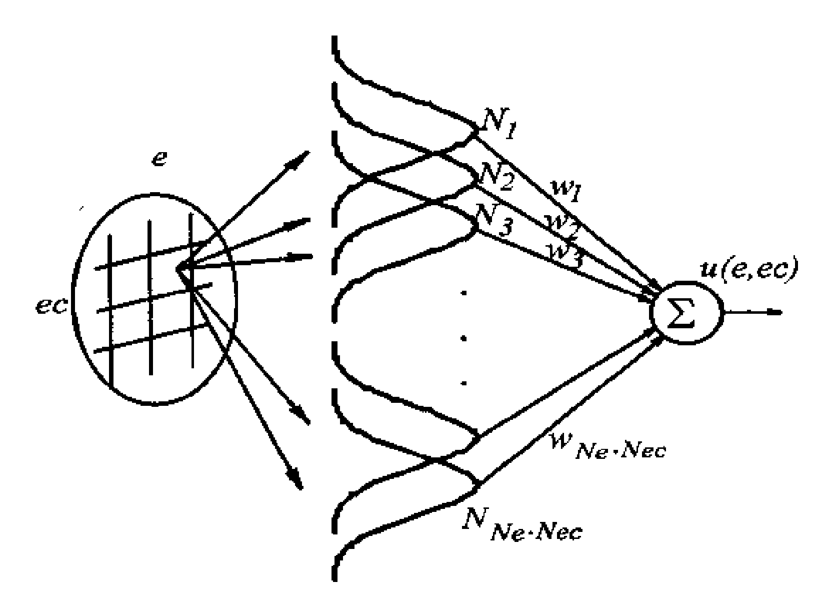


图 12.3 B 样条神经网络结构图

②根据 B 样条工具包计算(12.18)式中的控制值（给定一组初始权值 $w_k(0)$）；

③将控制器与被控过程组成闭环控制系统，并计算闭环系统的输出值，将其与期望值的误差平方和作为性能指标（适配值）；

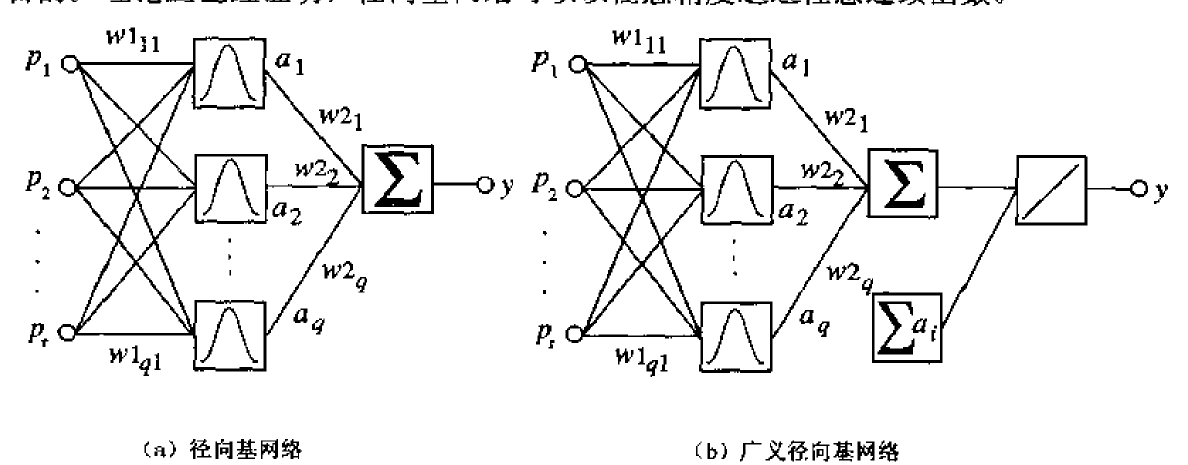
④根据计算出的适配值，对 B 样条模糊控制器中的权值 w_k 进行复制、交换和变异的遗传操作，以不断地优化适配值。在实际设计过程中，种群数取为 40，进化代数为 100。变异系数是随着进化代数的增加从 0.05 增加到 0.12。

12.6 径向基函数神经网络

对于直接采用前向网络实现模糊系统所存在的训练时间长，存在局部极小值问题，径

向基网络（Radial Basis Function Networks，简称 RBF 网络）在很大程度上解决了上述问题。RBF 网络不存在局部极小值问题，它不仅具有全局逼近性能，而且具有最佳逼近性能，同时其训练方法快速易行，为解决模糊神经网络的问题提供了新的思路。

在第 6 章中已经详细介绍了 RBF 网络，它的结构图如图 12.4 所示。从网络结构上看，RBF 网络含有一个隐含层，并且具有标准全连接的前向网络。隐含层中采用径向基函数——一种高斯型指数函数，而输出层采用线性激活函数，由此会使人感觉似乎 RBF 网络就是一个具有径向基函数的 BP 网络。实际上 RBF 网络是与 BP 网络完全不同的网络，其原因是：①它不是采用 BP 算法，即误差的反向传播法来训练网络权值的；②其训练的算法不是梯度下降法。虽然是两层网络，径向基网络的权值训练是一层一层进行的。在对径向基层中的权值进行训练时，采用的是无教师训练。网络训练的目的是使 $w1_{qj}=P_j^q$。由于径向基函数在将其输入放置在函数原点时输出为 1，而对其他不同的输入值的响应均小于 1，所以权值设计的目的地是将每一组输入值 P_j^q 作为一个径向基函数的原点，而权值 $w1$ 代表径向基函数中心的位置，通过令 $w1_{qj}=P_j^q$ 使每一个径向基函数只对一组 P_j^q 响应，从而迅速辨识出 P_j^q 的大小。然后进行输出层的权值设计。由于输出层是线性函数，网络输出是径向基层输出的线性组合，从而很容易地达到了从非线性输入空间向输出空间映射的目的。理论上已经证明，径向基网络可以以任意精度逼近任意连续函数。



（a）径向基网络　　　　（b）广义径向基网络

图 12.4 径向基网络结构图

值得指出的是，由于径向基网络的权值算法是单层进行，并且采用每一个 $w_{qj}=P_j^q$，所以虽然从网络结构图中看上去网络是全联接，实际上工作时网络是局部工作的，即对每一组输入，网络只有一个神经元被激活。其他神经元被激活的程度可以忽略。所以，径向基网络是一个局部逼近的神经网络。正是因为如此，网络的训练速度要比 BP 网络快上 1~2 个数量级。这是径向基网络的最大优点。不过径向基网络隐含层的神经元往往比较多，从上面的分析来看，它的数量一般情况下应与被训练的输入数组 Q 相等。当训练数组很多时，采用径向基网络可能不大容易被人接受。

对此缺点采用的改进算法是：在满足误差目标的前提下，尽量减少径向基层中的神经元数。具体的做法是在训练时从一个节点开始训练，通过检查误差目标使网络自动增加径向基层中神经元的节点，每次循环后用使网络产生最大误差所对应的输入矢量产生一个新的径向基层的神经元，然后检查新网络的误差，重复此过程直到达到误差目标为止。每增加一个径向基层的神经元，相当于增加了一条模糊规则，从而可以达到在线调整模糊规则的目的。

从网络结构上看，每个径向基层神经元的输出都是在每组输入值作用下的结果，与图 12.1 所表示的具有模糊逻辑推理的神经网络结构相比，RBF 网络中的径向基网络就起到了隶属函数的作用，径向基层就完成了相当于图 12.1 中的输入层加上中间层所完成的模糊推理功能。由于 RBF 网络中的径向基层是全连接，所以通过一层网络，对所有的输入变量都产生了作用，即让每一个输入变量在每一个径向基上都有一个输出。径向基层的输出即为所有输入变量共同作用的综合效果，其值为一组 0～1 之间的数，等价于模糊系统中从前提到结论所获得的所有模糊规则数。由此可见，RBF 网络用一个隐含层起到了普通模糊神经网络两层的作用。由于层数的减少，简化了结构，而且因其训练是两层分别进行，必然能够提高训练网络的速度。另外，径向基网络输出层的激活函数是线性函数，所以其网络输出应为 $Y = AW2 = \sum a_i w2_i$，这与模糊神经网络的表达式（12.6）完全相同。这说明，一个径向基网络的表达式所代表的就是一个模糊神经网络系统。

另外，与 B 样条网络相比较可以发现，径向基网络与 B 样条网络的表达内容完全相同，不同之处仅在于前者用径向基函数表示隶属函数，而后者的隶属函数是 B 样条基函数。

为了得到更加平滑的解模糊数值，可在普通的径向基网络输出后加上一个求加权平均值的过程，如图 12.4（b）所示，它比图 12.4（a）中多了一个除法运算符号“/”。这时的网络输出为：

$$y = \frac{\sum a_i(p) w2_i}{\sum a_i(p)} \tag{12.19}$$

此网络可称为广义径向基网络（或泛化回归网络）。

因为 a_i 的数值为 0~1 的指数型函数输出值，仔细观察（12.19）式，可以发现（12.19）式的形式与（12.3）式采用质心法对模糊集合进行解模糊的模糊系统的输出方式完全相同，只是在广义径向基网络中采用的是径向基函数，而模糊系统中是某一形状的隶属函数。正是由于这种径向基形状函数的作用，使广义径向基网络和模糊系统都具有相同的对局部输入上的激励在一个小的接受域内产生中心加权响应的机理。两者虽然是基于不同的原理发展而来的，但是都达到了同一目的，只是广义径向基网络具有较快的收敛速度，而模糊系统能够反映出更多数据的物理特性。

完全使广义径向基网络与模糊系统等价，还必须满足下列条件：

①RBF 网络和模糊系统必须采用相同的方法（即或加权平均法或加权求和法）产生其输出；

②RBF 网络中接收域的神经元个数等于模糊系统中的“如果—那么”的模糊规则数；

③RBF 网络中的每一个径向基函数等于模糊系统中组成某个模糊规则的多维隶属函

数的前提部分。达到这一目的方式之一是在模糊规则中采用与 RBF 网络具有相同的偏差的高斯型隶属函数，并应用点积来产生规则强度，从而使这些高斯型隶属函数的乘积变成一个多维高斯函数——RBF 网络中的一个径向基函数；

④RBF 网络与模糊系统应当对应有相同的响应函数，即它们应当具有相同的常数项（对于 $w2$ 取常数时，对应零阶 Sugeno 模糊推理系统，当取 $w2=c_0+c_1p_1+c_2p_2+\cdots+c_np_n$（$c_i$ 是待确定常数)时，对应于一阶 Sugeno 模糊推理系统）。

综上所述，对于模糊系统的实际应用，采用广义径向基网络不仅在隶属函数中心位置、宽度上，而且对模糊规则同时给予优化，另外具有相当快的收敛速度。其不足是在一般情况下需要较多的隐含层的神经元数。不过从模糊推理过程可以得知，既然模糊标记数和规则数体现出模糊系统的精度，所以在需要较高精度时，神经元数目较多是理所当然的，更何况 RBF 网络对神经元数目还可以自动寻优。

12.7 小　结

神经网络具有多种结构和学习算法，模糊逻辑推理也具有多种形式，本章从多方面研究了在用神经网络描述模糊控制的过程中，不同模糊神经网络之间的等价特性，以达到根据模糊推理规则来构造网络结构，同时利用神经网络的学习能力进行复杂的模糊推理，提高运算速度，达到对权值自动寻优的目的。通过对不同结构及方式的模糊神经网络系统关系的分析与对比，更加深入地揭示了各种网络的优缺点，有利于更好地选择和设计网络。

第 13 章 模糊神经系统的应用

13.1 基于 ANFIS 的非线性电机系统的建模

80 年代中期以来，随着人工神经网络研究的复兴，在许多领域取得了良好的应用效果。非线性系统的建模、辨识与控制是其中的重要应用方向。研究表明，对于非线性系统而言，采用传统的分析方法只能面向特定的应用，而不存在一种普遍适用的方法。人工神经网络以其出色的非线性映射逼近能力以及自学习能力为非线性系统的建模提供了强有力的工具。此外，随着对模糊推理系统研究的发展，使其不仅能够利用专家的语言知识，还可以根据给定的数据调整参数以获得良好的模糊模型。近年来，如何将模糊系统的知识表达能力与神经网络的学习能力结合起来，是备受注目的课题之一。

本章将着重介绍采用基于 Takagi-Sugeno 模型而建立的自适应神经模糊推理系统（简称 ANFIS），并同时对一个非线性电机系统进行建模，其主要目的是进行以下几项工作：①建模方法的对比。将采用 ANFIS 和反向传播网络（简称 BPNN）对同一系统分别建模，对各自的训练速度和建模精度进行对比；②采用 ANFIS 训练所建模型与实际系统之间进行输入/输出数据的验证以及泛化性能的检验。

本章的结构安排如下：首先简要介绍 ANFIS 的结构及参数调整的算法，然后进行采用 ANFIS 建模的实例设计与分析并给出对比结果，最后给出小结。

13.1.1 ANFIS 的结构

由Jyh-Shing R. Jang 提出的自适应神经模糊推理系统，是一种基于Takagi-Sugeno模型（或简称Sugeno模型）的模糊推理系统。研究表明，当输入模糊集采用非梯形/非三角形的隶属函数时，Sugeno模糊系统比玛达尼模糊系统更经济，即需要的模糊规则及输入的模糊集的个数较少。

一个具有两条规则的简单的Sugeno模糊系统ANFIS的结构如下：

$$\begin{aligned} &\text{if } x \text{ is } A_1 \text{ and } y \text{ is } B_1 \text{ then } f_1 = a_1 x + b_1 y + c_1 \\ &\text{if } x \text{ is } A_2 \text{ and } y \text{ is } B_2 \text{ then } f_2 = a_2 x + b_2 y + c_2 \end{aligned} \tag{13.1}$$

对应的 ANFIS 结构如图 13.1 所示。

需要指出的是，在图 13.1 中，节点间的连线仅表示信号的流向，没有权值与之关联；方形节点表示带有可调参数的节点，圆形节点表示不带有可调参数的节点。从图 13.1 可见，只有第 1 层和第 4 层中有可调参数。

各层的功能如下：

第 1 层：将输入变量模糊化，输出对应模糊集的隶属度，其中一个节点的传递函数可以表示为：

$$O_i^1 = \mu_{A_i}(x) \tag{13.2}$$

图 13.1 ANFIS 结构图

根据所选择的隶属函数的形式，可以得到相应的参数集，称为条件参数。例如，通常使用的高斯隶属函数：

$$\mu_{A_i}(x) = \exp\left(-\frac{\|x-d_i\|^2}{\sigma_i^2}\right) \tag{13.3}$$

则条件参数集为所有$\{\sigma_i, d_i\}$的集合。

第 2 层：实现条件部分的模糊集的运算，输出对应(13.1)式的每条规则的适用度，通常采用乘法：

$$O_i^2 = \omega_i = \mu_{A_i}(x) \times \mu_{B_i}(y) \tag{13.4}$$

第 3 层：对各条规则的适用度进行归一化处理。

$$O_i^3 = \overline{\omega}_i = \frac{\omega_i}{\omega_1 + \omega_2} \tag{13.5}$$

第 4 层：每个节点的传递函数为线性函数，表示局部的线性模型，计算出每条规则的输出：

$$O_i^4 = \overline{\omega_i} f_i = \overline{\omega_i}(a_i x + b_i y + c_i) \tag{13.6}$$

由所有$\{a_i, b_i, c_i\}$组成的参数集称为结论参数。

第 5 层：计算所有规则的输出之和：

$$O_i^5 = f = \sum \overline{\omega_i} f_i = \frac{\sum_i \omega_i f_i}{\sum_i \omega_i} \tag{13.7}$$

从以上网络的输入和输出之间的关系可以看出，图 13.1 所示的网络与（13.1）式所表示的模糊推理系统完全等价。模糊推理系统的参数学习可归结为对条件参数（非线性参数）

与结论参数（线性参数）的调整。

13.1.2 混合学习算法

对于所有参数，均可以采用基于梯度下降的反向传播算法来调整参数。这里对 ANFIS 参数的训练与确定，采用一种混合算法，其目的是为了提高学习的速度。混合算法中条件参数仍采用反向传播算法，而结论参数采用线性最小二乘估计算法来调整参数。

根据图 13.1 所示 ANFIS 系统，系统输出可写为：

$$\begin{aligned} f &= \sum_{i=1}^{2} \overline{\omega_i} f_i \\ &= \sum_{i=1}^{2} \overline{\omega_i} (a_i x + b_i y + c_i) \\ &= (\overline{\omega_1} x) a_1 + (\overline{\omega_1} y) b_1 + (\overline{\omega_1}) c_1 + (\overline{\omega_2} x) a_2 + (\overline{\omega_2} y) b_2 + (\overline{\omega_2}) c_2 \end{aligned} \tag{13.8}$$

混合学习算法的步骤如下：

每一次迭代中，输入信号首先沿网络正向传递直到第 4 层，此时固定条件参数，采用最小二乘估计算法调节结论参数；然后，信号继续沿网络正向传递直到输出层（即第 5 层）。此后，将获得的误差信号沿网络反向传播，从而可调节条件参数。

采用混合学习算法，对于给定的条件参数，可以得到结论参数的全局最优点，这样不仅可以降低梯度法中搜索空间的维数，通常还可以大大提高参数的收敛速度。

13.1.3 非线性电机系统建模

下面，将对一个具有非线性摩擦力影响的直流电机系统进行建模。用来采集输入/输出信号的实际控制系统包括一台 Pentium 200 计算机、一块内置于计算机的 12 位 A/D，D/A 转换板、PWM 功率放大电路、直流力矩电动机以及用于速度反馈的直流测速发电机。模拟电压输入范围与输出控制电压范围均为[-5, 5]伏，模数转换后的数字量范围均为[-2048, 2048]，为了方便起见，输入和输出单位均采用数字量。被控系统的模型包括除计算机外的所有部件的集合。在 5 毫秒的采样周期下，输入信号持续 10 秒，即 2000 个采样周期。为了使所建模型对各种不同的输入信号都具有很好的泛化能力，训练输入信号采用一个多频率分量正弦的复合信号：

$$u(k) = 350\sin\frac{k\pi}{1000} + 300\sin\frac{k\pi}{500} + 400\sin\frac{3k\pi}{1000} + 340\sin\frac{k\pi}{125} + 430\sin\frac{3k\pi}{200} + 320\sin\frac{k\pi}{25} \tag{13.9}$$

将（13.9）式的信号作为实际系统的输入，获得电机的真实输出（即转速信号）。实际电机的输入/输出如图 13.2 所示。

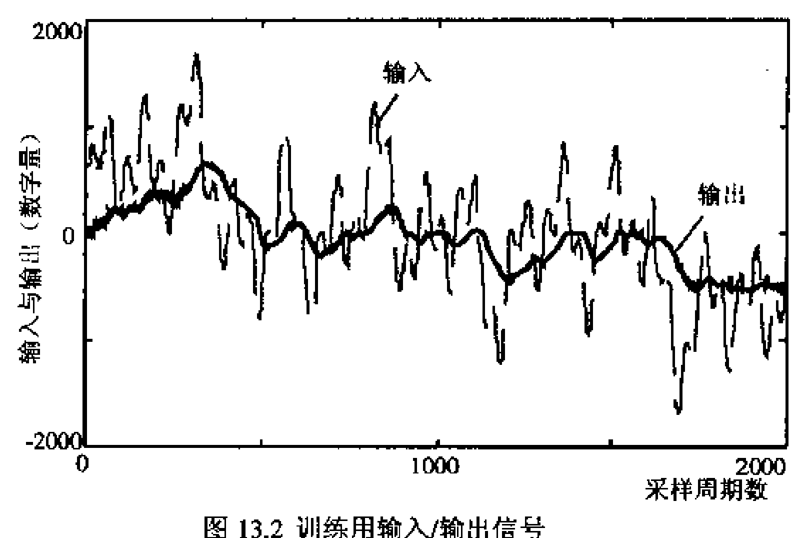


图 13.2 训练用输入/输出信号

13.1.4 基于 ANFIS 的建模

在 13.1.2 节中，已经介绍了 ANFIS 的结构及算法，但未提及 ANFIS 中参数的初始化问题。通常结论参数初始值为零，故初始化问题主要在于条件参数。条件参数的初始化，决定了输入空间的分割，从而决定模糊规则的数目。通常输入空间的分割可以采取平均分割法或采用模糊聚类方法。

首先，假设电机模型为一阶系统，即输入/输出关系式可以表示为：

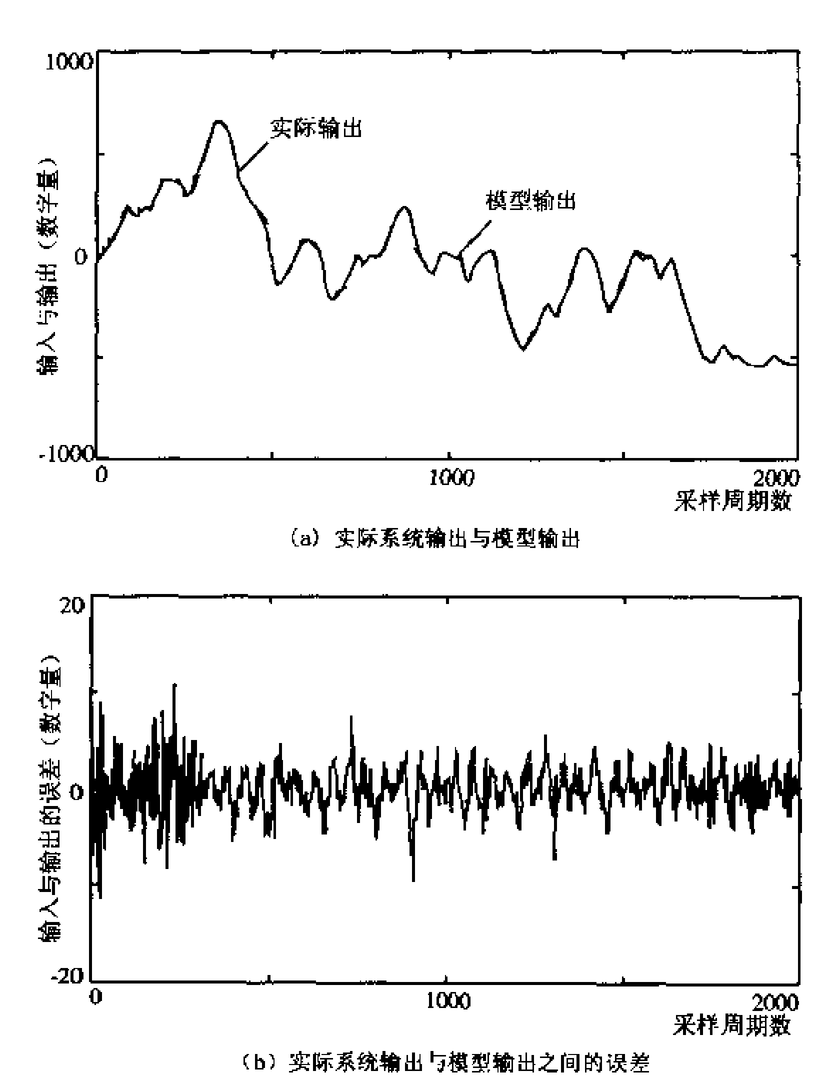
$$y(k)=f\big(y(k-1),u(k-1)\big) \tag{13.10}$$

我们采用平均分割法，对 $y(k\text{-}1)$与 $u(k\text{-}1)$均将各自的输入区间等分为 3 部分，模糊隶属函数均选用高斯型。这样，两个输入总共形成 9 种组合，即存在 9 条模糊规则。根据前面的分析可知，条件参数共有 12 个（3*2+3*2），结论参数共有 27 个（9*3）。由此可构造一个输入为 $u(k\text{-}1)$和 $y(k\text{-}1)$，输出为 $y(k)$，模糊标记数为 3，即 $i=3$ 的 ANFIS 模糊神经建模系统。实际系统的输出与模型输出之间的均方差 MSE 作为网络训练的指标。经过 100 次迭代，ANFIS 与实际系统之间的对比结果如图 13.3 所示，网络训练后的均方误差 $MSE=4.9854$。

作为对比，同时采用一个 2-10-1 的反向传播神经网络（BPNN）对同样的数据组进行建模。此时网络共有 41（$10*\omega_1*2+10*b1+10*\omega_2+b2$)个参数（而 ANFIS 只有 39 个参数）。对于 BPNN 的训练，采用收敛速度和训练效率相当好的 Levenberg-Marquardt 算法。这一算法收敛速度较普通的梯度下降法快得多，且迭代次数少很多。经过 1000 次迭代，BPNN 的训练结果的均方误差 $MSE=5.1516$。而采用 ANFIS 建模，达到相同的精度，迭代次数只需前者的十分之一。

通过实验发现，ANFIS 第一次迭代后即可得到很好的结果；而采用 BPNN 在前几十次迭代中，所具有的误差都一直比较大。虽然其 MSE 随迭代次数的增加有显著下降，但其效果仍不如 ANFIS。这是因为 ANFIS 的条件参数在初始化时采用的是平均分割法，相

当于把输入空间分成多个区域，与 BPNN 参数初始化的随机性相比，这一方法更加合乎情理；此外，ANFIS 实际上是由多个局部映射组合而成的，并且每个局部映射的参数（即结论参数）采用线性最小二乘估计算法，这就使得它的收敛速度比采用反向传播算法的 BPNN 快得多。总之，ANFIS 从初始化和学习算法两个方面保证了比 BPNN 更快的收敛速度。



(a) 实际系统输出与模型输出

(b) 实际系统输出与模型输出之间的误差

图 13.3 ANFIS 与实际系统之间的对比结果

13.1.5 辨识模型的验证

本节对于所辨识出的网络模型进行验证。首先，同样采用一个在幅值和频率上都有变化的多频率分量正弦信号作为测试信号：

$$u(k)=350\sin\frac{k\pi}{1000}+350\sin\frac{k\pi}{500}+450\sin\frac{k\pi}{200}+300\sin\frac{k\pi}{100}+450\sin\frac{k\pi}{50}+300\sin\frac{k\pi}{20} \quad (13.11)$$

对所建的非线性模型进行测试。测量到的实际系统与模型输出的结果如图 13.4 所示，均方误差为 $MSE=6.5836$。

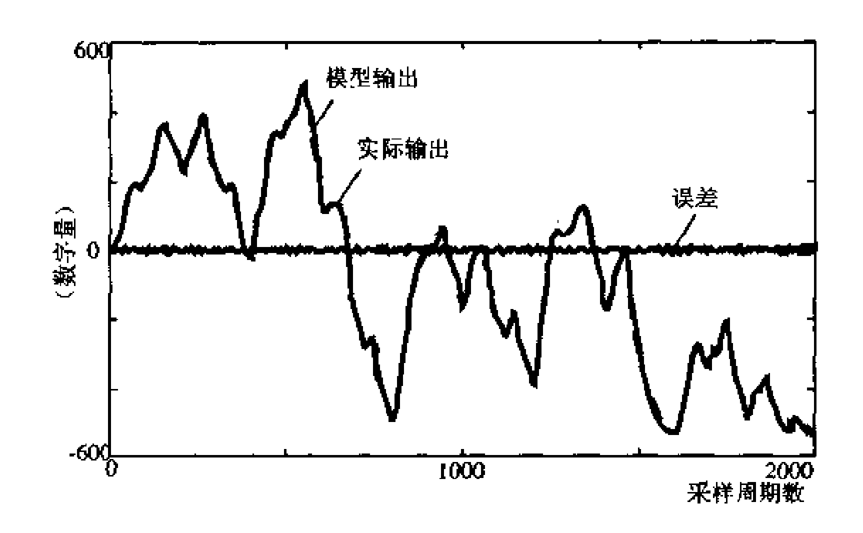


图 13.4 系统及模型分别对多频输入信号的响应曲线

其次，测试被建模系统所具有的两种典型的非线性特性：饱和与死区。分别采用两个单频率的正弦信号作为测试信号（见（13.12a）式与（13.12b）式）输入给所建模型，同时输入给实际系统，从实际系统及所建测试模型所获的输出及输入信号分别如图 13.5 和图 13.6 所示。

$$u(k)=1000\sin\frac{k\pi}{1000} \tag{13.12a}$$

$$u(k)=400\sin\frac{k\pi}{1000} \tag{13.12b}$$

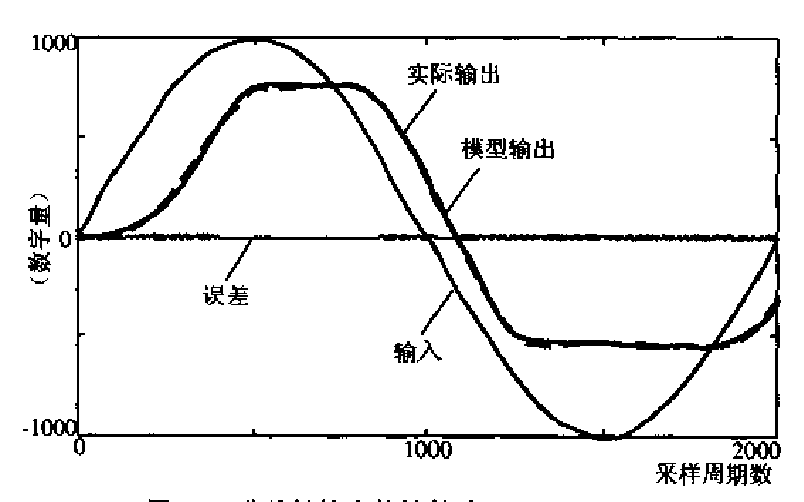


图 13.5 非线性饱和特性的验证（MSE=5 .461）

从实际系统输出与所建模型的输出可以看出，在 2000 个采样点的数据中所具有的误差平方和均在 7 以内，可见其精度之高。从图中也可以看出实际输出与模型输出几乎重合为一条曲线。

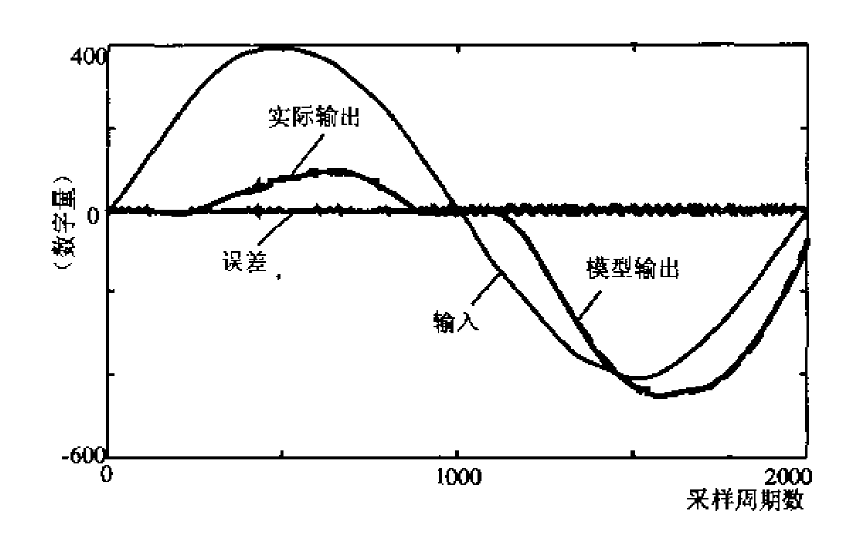


图 13.6 非线性死区特性的验证（MSE=1.9406）

由以上测试可以看出，所建立的 ANFIS 模型不仅可以很好地适应幅值与频率的变化，而且能够很好地包含电机系统的非线性特性。所建立的模型在训练信号所在的范围内对幅值与频率的改变有很强的适应性，达到了动态精确建模的目的。

13.1.6 小结

本节采用 ANFIS 对一个非线性电机系统建模，获得了一个适应范围很大的全局非线性动态模型。通过与 BPNN 的比较，说明 ANFIS 的主要优点在于收敛速度很快，收敛时间只需 BPNN 的十分之一，而当面对复杂系统时，建模所需参数急剧增多，收敛速度显得至关重要，此时 ANFIS 则比 BPNN 显示出更大的优越性。同时，在对实验结果的分析中，阐明了 ANFIS 从初始化和学习算法两个方面保证了比 BPNN 更快的收敛速度的原因。此外，ANFIS 本质上是一个基于网络结构的模糊推理系统，使其具有比 BPNN 更好的知识表达能力。在实际应用中，ANFIS 为非线性系统的建模、辨识以及时间序列分析提供了有力的工具。

13.2 用自组织竞争网络优化模糊神经网络的结构

神经网络与模糊逻辑控制相结合的研究已受到人们的广泛注意，通过神经网络所实现的模糊逻辑控制器的结构具有模糊逻辑推理功能，同时网络的权值也具有明确的模糊逻辑意义。以此方式，用各自的优点弥补对方的不足，可以达到最佳的设计效果。不过，不论采用何种训练方法，所获得的权值，仅仅是模糊神经网络结构在事先固定的情况下的最优，代表模糊标记(即：正小、负大等等)的权值的数目是由设计者在训练之前就已选定的，而这个数目为多少经常是根据设计者的经验或是由误差精度的要求来确定的，很明显很难选得一个最优值或称最小值：若此模糊标记的数选得太小，则很难达到期望的目标；若选得

太大，网络输出层的模糊控制规则数将以指数形式增加，这必然造成庞大的网络结构，给权值的训练及网络的实现均带来不便。

本节给出一种解决此问题的设计新方法。为了获得同时具有最佳结构和参数的模糊神经网络 (Fuzzy-Neural Network, 简称 FNN)，运用自组织竞争神经网络 (Self-Organization Competition Neural Network, 简称 SCNN) 优化网络结构。其设计过程分成三步：

首先，设计一个具有较多权值的模糊神经网络，并确定其模糊标记数(比如选 7 个)，然后，通过训练优化其权值；

第二步，选取模糊神经网络中代表模糊隶属函数的权值(包括偏差)作为自组织竞争神经网络的输入矢量。通过该网络的竞争与训练过程，将这些输入矢量的相同的类别自动组合成若干组，竞争后所获得的组合数将成为模糊神经网络的最佳模糊标记数，以此方式将模糊标记的数目减少到最小；

最后，重新训练由第二步竞争出的最小模糊标记的 FNN 权值。

13.2.1 自组织竞争神经网络

自组织竞争神经网络是基于生物结构和现象形成的，它能够对输入模式进行自组织训练和判断，将其最终分为不同的类型。SCNN 权值的修正采用的是 Kohonen 规则，该学习规则仅对与竞争层获胜节点相连的权值进行修正，修正的目的是使输入矢量与权值之间的误差趋于 0，以致使网络的权值在竞争与训练结束后，能够代表输入矢量，并用输出值对应其种类。输入矢量之间的相似性，是由事先选定的偏差值 b 来控制的。竞争层的神经元数（节点）r 是由设计者确定的，它代表输入矢量可能被划分的种类数。

考虑输入矢量 $X_{m\times n}$，SCNN 的训练过程如下：

①计算：计算每一个竞争层节点的加权输入总和 u_i；

$$u_i=\sum_{j=1}^{m} w_{ij}x_{jq}+b,\qquad i=1,2,\ldots,r;\qquad q=1,2,\ldots,n \tag{13.13}$$

②竞争：竞争层中加权输入总和值为最大的节点为获胜节点，假定获胜节点为第 l 个，$l\in[1,r]$，则有：

$$u_l=\max(u_i,i=1,2,\ldots,r) \tag{13.14}$$

对于输出节点 i 有下式成立：

$$y_i=\begin{cases}1, & \text{如果,}\quad i=l\\ 0, & \text{其他}\end{cases} \tag{13.15}$$

③修正权值：修正与获胜节点 l 相连的权值，其他权值保持不变，根据 Kohonen 规则，有：

$$\Delta w_{lj}=\eta(x_{jq}-w_{lj}),\qquad w_{lj}=w_{lj}+\Delta w_{lj} \tag{13.16}$$

其中，η 是学习速率，x_{jq} 为经过归一化处理的输入矢量：

$$x_{jq} = \frac{x_{jq}}{\sqrt{\sum_{j=1}^{m} x_{jq}^2}}, \qquad q = 1,2,\ldots,n \tag{13.17}$$

④循环：再次计算新的加权输入矢量和，重复进行竞争与权值的修正过程，直到达到预选定的循环总数 N 时停止训练。

因为只有与获胜节点相连的权值才能得以修正，所以随着竞争与训练过程的重复，那些被修正的权值则越来越趋向于它们的输入矢量，此趋势从方程(13.16)中也可以推导出：当 $w_{lj} = x_{jq}$ 时，$\Delta w = 0$。而那些获胜节点，在当另一个相似的输入矢量出现时（只要此输入矢量与已获胜的输入矢量之间的偏差小于设定偏差值 b），仍然能够获胜。而当一个具有大于 b 偏差的输入矢量出现时，此输入节点不能取胜，但定有另一个节点能够获胜，这个输入矢量则属于另一类。SCNN 的整个训练过程实际上就是一个将输入矢量划分成若干类的过程，使相似的输入变成一类，相同类具有相同权值，从而降低了输入矢量的种类数。

13.2.2 具有最佳结构与参数的模糊神经网络控制器的设计

为了展现模糊神经网络中 SCNN 的作用，在此将所提出的方法应用到一个模糊神经网络（FNN）控制器的设计中，其控制器的设计是用来控制一个具有高度非线性摩擦力影响的直流伺服电机的速度跟踪系统，控制目的是为了消除非线性摩擦力所引起的死区，并精确地跟踪参考输入。

1) FNN 控制器的设计

首先，设计一个具有 7 个模糊标记，即具有 49 个控制规则的 FNN 控制器，网络结构如图 13.7 所示。可以看出它是一个具有输入层、中间层和输出层的三层神经网络。在功

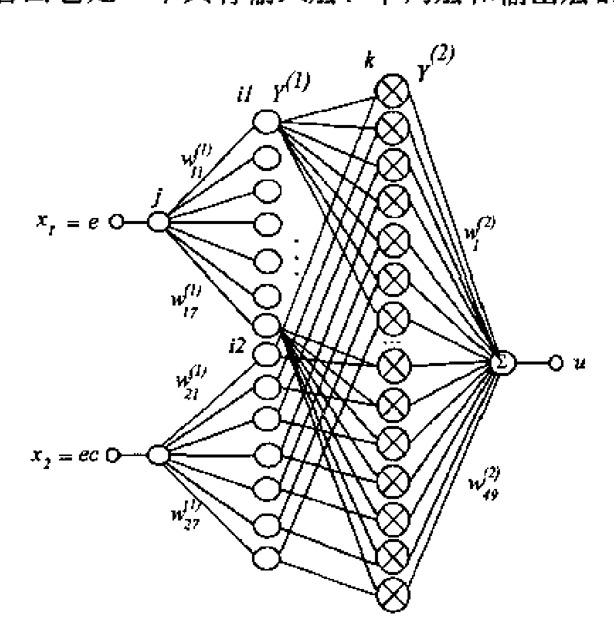


图 13.7 模糊神经网络控制器的结构图

能上，该网络的三层节点是严格对应于模糊逻辑控制的模糊化、规则推理和逆模糊三个步

骤的，因而具有明确的模糊逻辑意义。首先，网络输入变量为误差 e 和误差变化 ec；其次，输入层节点的转移函数代表的恰为模糊变量的隶属函数。该层的权重值 $w_{ij}^{(1)}$ 和 θ_{ij} 的不同也就意味着变化多端的隶属函数的形状和位置。该层的输出 $Y^{(1)}$代表的就是模糊化的结果—隶属度。再者，中间层是将模糊化得到的隶属度两两相乘的功能。中间层的输出代表着模糊规则的规则强度，将这些强度传递给下一层就可以进行逆模糊了。最后，输出层的各个权重值 $W^{(2)}$代表了模糊规则，根据重心法的逆模糊方式，只要将它们作为权重值与输入，即规则强度加权求和，输出即为模糊控制的输出量。

总的来说，FNN 的输入/输出关系可以归结如下：

$$X=[eec]^T \tag{13.18}$$

$$y_{ij}^{(1)}=\exp[-(w_{ij}^{(1)}\cdot x_j+\theta_{ij})^2] \tag{13.19}$$

$$y_k^{(2)}=y_{i1}^{(1)}\cdot y_{i2}^{(1)}=\exp\{-[(w_{i1}^{(1)}\cdot x_1+\theta_{i1})^2+(w_{i2}^{(1)}\cdot x_2+\theta_{i2})^2]\} \tag{13.20}$$

$$u=W^{(2)}\cdot Y^{(2)}=\sum_{k=1}^{49}y_k^{(2)}\cdot w_k^{(2)} \tag{13.21}$$

其中，$j=1,2$；$i=i1,i2$ ； $i1,i2=1,2,\ldots,r1$；$k=1,2\cdots,r1\times r1$；$r1$ 为模糊标记数。

为了获得最优网络权值以及期望的闭环控制精度，采用间接法对权值进行训练，即将控制器与被控过程串联起来，并形成一个负反馈控制回路，将参数输入与闭环回路的输出之间差的平方和作为目标函数，以保证总体最优的质量。另外考虑到反向传播法训练权值的费时以及有陷入局部极小值的不足，可采用改进的遗传法来优化权值。

2) 采用 SCNN 优化模糊标记数

对于上述具有 7 个模糊标记的模糊神经网络，在总共 77 个参数中，有 28 个权值和偏差代表不同输入的隶属函数。它们可以被分为两组：一组由误差变量 e 的 7 个权值和 7 个偏差组成，另一组包含误差的变化变量 ec 的 7 个权值和 7 个偏差值。现在将这两组为 SCNN 的输入矢量竞争层的节点数选为 5，最大循环数 $N=500$，相似度偏差 $b=-0.9$。在对输入矢量归一化处理后，开始进行竞争和权值的训练，竞争学习规则如节点所述，训练目标是使相似的输入矢量聚成同一类型，以达到减少输入矢量数组的目的。所有的设计与训练程序均由 MATLAB 及其中的神经网络工具包完成。

图 13.8 给出了竞争和训练之后的输入与 SCNN 权值矢量的分布图。其中“线”表示输入矢量，“+”代表权值矢量值，因为参数被归一化处理，所以输入矢量具有单位模数，所有的输入矢量值均落在单位圆周上。从图 13.8 可以看出，通过竞争后，由 4 个权值代替了 7 个输入矢量，事先设置的 5 个权值矢量中还多出一个。

按照一般的概念，通常模糊标记的数目的选择是单数，且最小数应当取 3：负、零和正，而此处在经过竞争训练后为什么对所研究的问题给出了 4 个标记数？为了回答这个问题，同时为了比较，对具有 3 个模糊标记数的模糊神经网络控制器也进行了训练设计，同样采用改进的遗传算法优化网络权值。不幸的是，不论如何训练网络，不能达到期望的误

差值，控制系统也不能跟踪输入信号，图 13.9（a）给出了训练后所得到的一种情况：能够很好地跟踪正向参考输入。而负向跟踪产生较大误差时的隶属函数曲线，图 13.9（b）和（c）分别为模糊标记为 4 和 7 时的隶属函数曲线。比较三者可以看出，不论选取几个模糊标记，总存在一条平坦的隶属函数曲线，此隶属函数是用来克服被控过程中存在的非

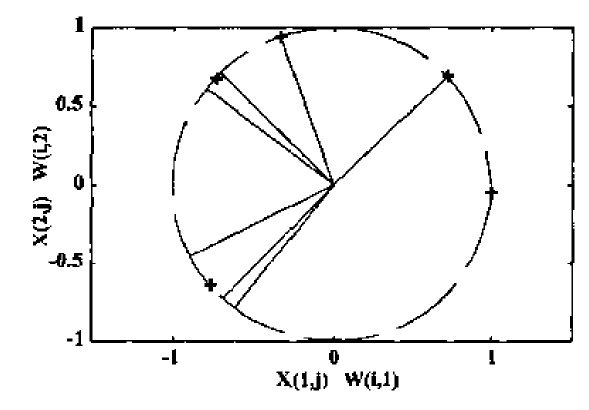
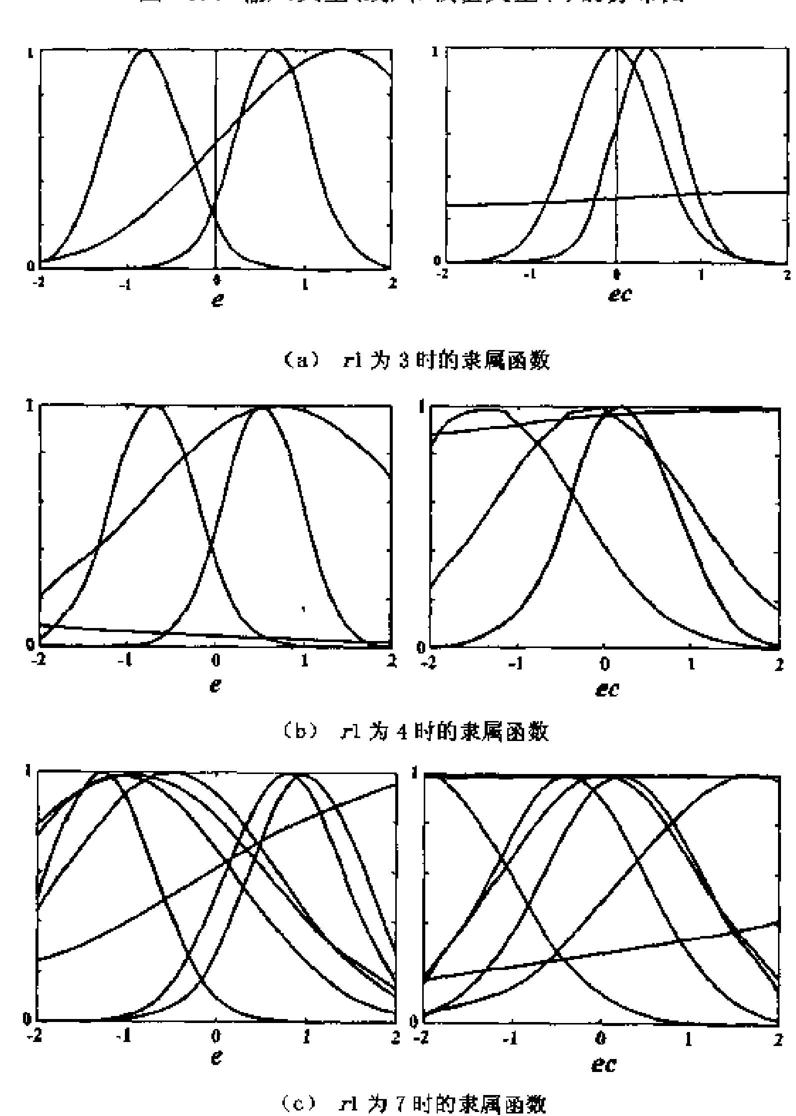


图 13.8 输入矢量(线)和权值矢量(+)的分布图



（a） r1 为 3 时的隶属函数

（b） r1 为 4 时的隶属函数

（c） r1 为 7 时的隶属函数

图 13.9 隶属函数曲线

线性摩擦力。当模糊标记数为 3 时，除了一个用于非线性补偿外，只剩下两个可以用来控制动力学特性，很明显是不够的，从图 13.9（a）中可以看出，*ec* 中只有一个正和零的隶属函数，然而没有表示负的隶属函数，所以系统不能很好地跟踪负向信号，而当模糊标记数为 4 时，如图 13.9（b）所示，一个用来消除非线性摩擦力，三个用来分别跟踪负、零和正向信号，以达到最佳控制效果，当模糊标记数多于 4，那么多余的隶属函数或出现在 3 个主要的隶属函数附近，或协助克服非线性，所有这些多余的“曲线”，均能通过 SCNN 的训练而被 4 条隶属函数代替。

在通过 SCNN 的竞争和训练后，获得最小的模糊目标的数目，然后，用 4 个模糊标记重新构成一个新的模糊神经网络，并可再次应用上述同样的过程训练出网络权值。

图 13.10 给出速度跟踪系统在模糊神经网络控制器作用下，对梯形输入信号的响应，其中，图 13.10（a）通过 SCNN 训练结果获得的具有 4 个以及原有 7 个模糊记标的模糊神经网络控制器的响应信号，图 13.10（b）为其响应误差信号，其中实线为具有 4 个模糊标记的 FNN 控制器的误差。从图中可以看出，两个控制系统具有完全相同的跟踪输入信号能力，这意味着：前者可以取代后者，同时也是最小数目和最简结构，因为当其数目小于 4 时，模糊神经网络控制器不能很好地工作。所以可以说，由通过 SCNN 的训练所获得的模糊标记数所组成的模糊神经网络结构及其训练出的权值具有最优结构与权值。

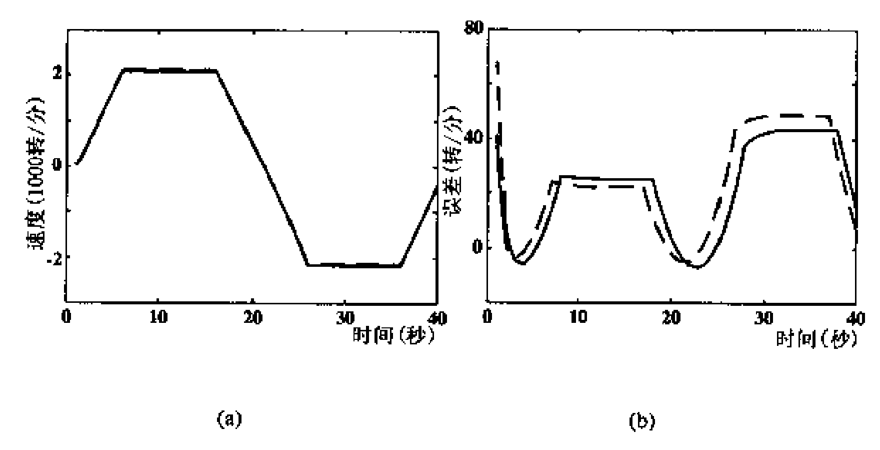


图 13.10 输入信号与不同情况下的控制系统响应(a)与误差信号(b)

13.2.3 小结

本章结合 SCNN 训练竞争出最佳模糊标记数，以及改进的遗传算法优化网络的权值，提出一种设计模糊神经网络最佳结构与权值的方法。竞争训练前后的隶属函数的对比以及控制效果的对比都证明了所提方法的有效性。

第14章 遗传算法

19世纪50年代，英国生物学家达尔文（C. R. Darwin，1809 ~ 1882)根据他对世界各地生活的考察资料和人工选择的实验，提出了生物进化论。自然选择学说是达尔文进化论的中心内容。1859 年达尔文发表《物种起源》巨著，提出了以自然选择为基础的生物进化论学说。

根据达尔文的进化论，生物发展进化主要有三个原因，就是遗传、变异和选择。遗传就是子代总是和父代相似，遗传性是一切事物所共有的特性，正是这种遗传性，使得生物能够把它的特性、性状传给后代，在后代中保持相似。现代遗传学研究表明，生物具有遗传性是生物都具有遗传的基础，遗传学的建立和发展有力推动了进化论，所以说，遗传是生物进化的基础。变异是指子代与父代有某些不相似的现象，即子代永远不会和父代完全一样。变异是一切生物所具有的共同特征，是生物个体之间相互区别的基础。引起变异的原因主要是生活条件的影响、器官使用的不同及杂交。生物的变异性为生物的进化和发展创造了条件。选择决定生物进化的方向，所谓选择是指保留和淘汰的意思。选择分为人工选择和自然选择，人工选择是在人为环境下，把对人有利的生物保留下来，对人体不利的个体被淘汰。自然选择就是指生物在自然界的生存环境中适者生存，不适者被淘汰的过程。世界上所有形形色色的生物，都是在自然选择的影响下，在悠久的岁月中形成的。自然选择的过程是通过生存斗争的过程来实现，生存斗争的结果，优存劣汰，这样就保存那些适应环境、有利于生存的变异。通过不断的自然选择，这些有利于生存的变异就遗传下去，积累起来，使变异越来越大，逐步产生新的物种。

总之，生物就是在遗传、变异与选择三种因素的综合作用过程中，不断地向前发展和进化。选择是通过遗传和变异起作用的，变异为选择提供资料，遗传巩固与积累选择的资料，而选择则能控制变异和遗传的方向，使变异和遗传向着适应环境方向发展。这样，生物就会从简单到复杂，从低级到高级不断地向前进化和发展。

自然选择学说能够正确地解释生物界的自然现象——多样性和适应性，这对于人们正确认识生物界具有重要意义。生物进化论揭示了生物长期自然选择的进化的发展规律，使科学家们从中受到了启迪，认识到进化论、自然选择过程蕴含着一种搜索和优化的先进思想，并将这种思想用于工程技术领域，发展出遗传算法，为解决许多传统的优化方法难以解决的优化问题提供了崭新的途径。

遗传算法（Genetic Algorithm，简称 GA）是建立在自然选择和自然遗传学机理基础上的迭代自适应概率性搜索算法。该算法最早是由美国的 J.H.Holland 教授 1975 年发表的论文“自然和人工系统的适配”中提出的一种模仿生物进化过程的最优化方法，它模拟了自然选择和自然遗传过程中发生的繁殖、交换、变异现象，它根据适者生存、优胜劣汰的自然法则利用遗传算子：选择、交叉、变异逐代产生、优选个体，最终搜索到较优的个体。具有不需要求梯度、能得到全局最优解、算法简单、可并行处理等优点，遗传算法已成功应用于各种复杂问题的优化中，在许多传统优化技术难以解决的场合更显示出其优越性。

80 年代以来已成功地应用在机器学习和复杂的函数优化等许多领域。近几年来，并行分布处理的发展使人们看到了遗传算法潜在的价值。

在将遗传算法应用于实际问题之前，首先必须将待优化的参数进行编码。一般的说，总是用二进制将参数编码成 0 与 1 组成的有限长度的字符串，该字符串称为染色体，其中的每个字符称为基因。但是也可根据实际问题的需要选择其他的编码方法。另外由于遗传算法利用群体中每个个体的优劣信息进行搜索，因而必须根据优化目标对每个个体进行评价，确定一个性能指标，该指标被称为适配度。遗传算法源于生物遗传学，其中很多术语是从遗传学中借用过来的。表 14.1 列出一些常用的遗传学术语与遗传算法术语之间的对应关系。

表 14.1 遗传学术语与遗传算法术语之间的对应关系

遗传学	遗传算法
染色体(chromosome)	字符串（或样本）个体(individul)
基因(gene)	每个字符串
代(generation)	种群
繁殖（reproduction）再生	选择、复制
交配(crossover)	交叉、交换
变异(mutation)	变异

14.1 遗传算法的基本特点

遗传算法有以下的基本特点：

①传统优化算法通常是直接处理函数和它们的控制变量，而遗传算法通过编码将优化问题的自然参数编码成有限长度数字串位，故遗传算法基本上不受函数约束条件（如连续、可导、单调等）特性的限制，能在极其广泛的问题求解过程中发挥作用。

②传统搜索法基本上是点到点的搜索方法。在多极点的搜索空间里常陷入局部值。而遗传算法则从由点组成的“群体”开始搜索，并行地“爬”过多个“山峰”，使陷入局域解的可能性大大减小。

③另外在遗传算法的繁殖进化过程中，这种点间的信息交换可明显加速遗传算法的进化过程。Holland 证明了在一个规模为 n 个的种群中，$O(n^3)$模式是有效的，即每一代遗传算法在处理 n 个串的模式同时，实际上有效地处理大约 n^3 个模式。这是一个非常重要的性质。Holland 称之为隐并行性。

④遗传算法在选择过程中，依据一定概率随机地选择个体是为了模仿自然界进化过程中的“适者生存”、“优胜劣汰”的竞争规则，使得适应环境的能力较强的个体拥有较多的再生机会，而这些有着较强生命力的解群在引导遗传算法的搜索方向方面可能包含较多有价值的信息。允许较低适应度的个体以较低的概率取得再生机会是考虑到总体质量较差的可行解中可能包含某些优秀的个别特性。所以遗传算法虽然以随机化方法来进行搜索，实际上是朝着有可能改进解的质量的搜索空间进行搜索，即为一种有导向的随机化搜索方法。

大多数传统搜索方法都需要使用较多的附加特性，如可微、连续、单调等，而遗传算法基本上不用搜索空间的知识和其他附加特性，而仅用适配度函数值来评估个体，并在此

基础上进行遗传操作。由于遗传算法的适配度函数不受连续、可微、单调等性质的约束，且其定义域可以任意设定，因此，与传统搜索算法相比，遗传算法具有更强的鲁棒性，能适用于更广泛的应用领域。

综上所述，遗传算法是一类针对优化问题编码空间的、具有导向的随机化优化搜索方法。其自身隐含着并进行性以及本质上的鲁棒性，是遗传算法区别于传统优化搜索方法的主要标志。

14.2 遗传算法的基本操作

遗传算法是一种对群体的操作，该操作以群体中的所有个体为对象。

1）选择（复制）操作

遗传算法的选择策略是对达尔文进化论中“自然选择”学说的核心思想——“适者生存，优胜劣汰”的简单模拟，即选择目标主要包含两个方面的内容：一是配种选择，二是生存选择。配种选择的目标是希望通过对交义配偶有倾向性的选择，产生适配度较高的后代；生存选择（又称种群选择、样本选择）的目标是想通过对种群的筛选，更新保存较优的样本，从而为进化创造较好环境条件。

选择操作的目的是为了从当前群体中选出优良的个体，使它们有机会作为父代为下一代繁殖子孙。判断个体优良与否的准则就是各自的适配度。个体的适配度越高，被选中的机会就越大。

实现选择操作的方式很多，其类型主要有以下几种：

（1）直接基于适配度的比例选择机制

①赌轮(Roulette wheel selection)选择方式；

②期望值模型(Expected value model)选择方式；

③随机竞争(Stochastic tournament)选择方式。

（2）间接基于适配度的非线性选择机制

①线性标定(Scaling)：$f(z)=az+b$ ；

②幂函数标定：$f(z)=z^b$；

③指数标定：$f(z)=e^{-bz}$ 。

（3）基于代沟(Generation)方式的种群选择机制

（4）基于小生境技术的种群选择机制

①基于预选择(Preselection)机制的小生境技术；

② 基于种群(Crowding)机制的小生境技术；

③基于共享(Sharing)机制的小生境技术。

常用的选择操作是采用和适配度成比例的概率方法来进行选择。具体地说，就是首先计算群体中所有个体适配度的总和$\sum f_i$，再计算每个个体的适配度所占的比例$f_i/\sum f_i$，

并以此作为相应的选择概率。

2）交叉（交换）操作

在自然生物界，基因重组是保持生物特性遗传的基本方法，也是获取大量遗传变异的最主要来源。遗传算法中的交叉操作可视为对生物遗传过程中基因重组的直接模拟，其任务是将两个配对个体相结合，通过基因及部分结构的随机交换和重新组合方式生成新的个体。一般来说，交叉算子若缺少“遗传”性，父代的优良品质就难以被继承，进化过程的历史信息将不能被有效地利用，有利的遗传变异也得不到逐渐积累并在种群中稳定下来，遗传算法的局域搜索能力和收敛性均将受到不利影响。

此外，过强的“变异”能力还有可能使遗传算法的随机搜索无法收敛。另一方面，交叉操作若缺少“变异”性，遗传变异库就难以被更新和丰富，这将使进化过程缺乏动力，使遗传算法跳出局域陷阱的能力大为降低，还可能使初始化产生丰富的遗传变异迅速趋向单调，遗传算法趋向不成熟收敛。

所以在设计交叉算子时，需要兼顾如下两个基本要求：

①交叉操作要有利于父串关键特征（模式）以及有利于变异的遗传和继承；

②交叉操作要有利于遗传变异库的更新和丰富。

简单的交叉可分两步进行：首先对种群中的个体某一个概率值进行随机配对；其次在配对个体中随机设定交叉处，被配对个体彼此交换部分信息。交叉操作类型一般有一点交叉(one-point crossover)、二点交叉(two-point crossover)、多点交叉(multi-point crossover)和一致交叉(uniform crossover)几种。

遗传算法的交叉过程可以看作是等位基因的竞争过程，对于两个父串的同型等位基因，因为无竞争可言，子串直接继承这些同类型等位基因是很自然的，而对于两个父串的杂型等位基因（一个是 0，另一个是 1），在难以分辨孰优孰劣的情形下，采取纯随机方式进行选择就不失为一种可行的方法。随机性归纳法式交叉操作方法就是基于这种考虑设计的，它直接继承两个父串的同型等位基因，而对杂型等位基因则按纯随机方式产生。显然这一交叉方法使子代继承了双亲的同型等位基因；对于双亲的杂型等位基因，“与/或”交叉方法采取了两种不同的“强调”策略；“与”运算强调 0 基因作用，而“或”运算则强调 1 基因作用。有趣的是，这种交叉方法与生物遗传学的显性现象相类似，“与”运算将 1 视为隐性基因，而“或”运算将 0 视为隐性基因。这种有效交叉操作的增多，有利于维护种群的多样性，加速遗传算法的进化过程，提高遗传算法的优化效率。

3）变异操作

在遗传算法中，变异操作的主要作用是防止重要基因的丢失，维护种群的基因型多样性。在生物进化过程中，变异概率是相当小的，约在 10^{-6} 数量级。在遗传算法中，无论从确保遗传算法的收敛性，还是从提高遗传算法的优化效率方面考虑，变异概率均不宜取得过大，否则遗传算法将出现随机搜索。

在较小变异概率下，变异操作仅使种群基因组成的基因型结构发生微量的变化（能引起种群基因组成的基因型结构发生重大变化的变异操作，往往不太可能是有利的）。但选择操作的方向性和交叉操作的遗传性，将使微小的、点点滴滴的有利变异能得到逐渐积累，并在种群中逐步扩散和稳定下来。所以，变异操作是十分微妙的遗传操作，它需要和交叉操作妥善配合使用，目的是挖掘群体中个体的多样性，克服有可能限于局部最优解的弊病。

变异算子的具体做法是把某一个体中的每一位按某一个概率进行取反运算，即由 1 变为 0 和由 0 变为 1 。同自然界一样，每一位发生变异的概率是很小的，遗传算法的搜索能力主要是由选择和交叉操作完成，而变异操作则保证算法能搜索到问题解空间的每一点，使算法具有全局收敛性。

遗传算法依据每个个体的适配度利用选择、交叉、变异算子对个体进行更新换代。选择算子模拟了生物中的自然选择现象。适配度越大，被选中的可能性越大，其子孙在下一代中的个数越多。具体的实现方法有竞赛选择、转轮式选择，以及无替代剩余随机采样选择等。交叉算子则以一定的概率交换从某一位置起两个个体的部分基因，它模拟了群体繁殖过程中的交配现象。通过交叉算子就有可能将个体中的优良基因组合在一起，使个体表现出良好的性能。变异算子模拟了遗传机理中的变异现象，它以较小概率改变染色体中的基因。在二进制编码中，变异算子以一定的概率将某一位置的 1 变成 0，或者将 0 变成 1。变异本身是一种随机搜索，然而与选择、交叉算子结合在一起，就能避免由于选择与交叉算子而引起的某些信息的永久性丢失，从而保证了遗传算法的有效性。

14.3 遗传算法的设计步骤

遗传算法以随机产生的一群候选解为开始，而每一解均被表示成字符串形式。通过使用遗传算子对这些字符串进行组合，这些候选解逐步朝着更好解的方向进化。这些遗传操作（如选择、交叉和变异）则分别模拟自然选择和自然遗传过程中发生的基因繁殖、交配和突变现象。在每一代对于给定问题，我们维持了一个数目 N 恒定的群体，通过计算各解的适配值 f，使这些解得到评价。根据各解的是适配值的大小，分配繁殖机会，适配值相对高的个体在下一代得到更多的繁殖机会，产生更多的后代，而适配值低的个体则产生的后代数目少，甚至被性能更好的后代个体所代替。被选个体又利用交叉、变异等遗传算子进行组合，形成新的一代。

在应用遗传算法求解具体问题时，主要考虑以下几个问题：

1）*参数编码*

由于遗传算法不能直接处理空间的数据，所以必须通过有效且通用的编码方法，将问题的可能解编码表示成有限位的字符串，成为遗传空间的基因型串结构数据。

对寻优参数编码。确定寻优参数和各参数的变化范围，将各寻优参数用无符号的二进制数表示。设某寻优参数 a 的变化范围是$[a_{min},a_{max}]$，若用 m 位二进制数 b 来表示，则 b 可由下式求得：

$$b=(2^m-1)(a-a_{\min})/(a_{\max}-a_{\min}) \tag{14.1}$$

再将所有寻优参数的二进制数串联成一个二进制的字符串 s，又称为样本。若有 r 个寻优参数，每个参数都用 m 位二进制数表示，则字符串 s 共有 $m\times r$ 位。

另外，还采用实数等方法进行编码。

2）*种群初始化*

确定遗传算法所使用的各参数的取值，如群体规模 n，交叉、变异等发生的概率。

若无先验知识，可随机产生 n 个字符串（样本），组成一个种群(Population)。

3）求各样本的适配值

根据编码方法以及问题的要求，设计一个适配度函数 f，用以测量和评价各解的性能。这个适配度函数实际上对应于最优化的指标函数。用每个样本对应的一组寻优参数计算其适配值(Fitness Value)，并按从优到劣的次序排列。

4）选择（Seletion）

求出各样本的适配度：

$$p_i = f_i / \sum f_i \ , \ i = 1, 2, \dots, n \tag{14.2}$$

对种群中各样本以优于 p_i 值的原则选择出来作为父母样本，并随机地两两配对，用交叉和变异的方法繁殖后代。

5）交叉（Crossover）

在父母样本 A、B 的字符串中随机地产生一个分段点，将分段点之后的子串互换，生成两个子女样本，如下列：

A = 1101/011　　A′= 1101/110

B = 0010/110　　B′= 0010/011

交换概率可定为 0.6 ～1.0。

6）变异(mutation)

在每个子女样本的字符串中随机地选择一位，将其数值求反（即 1 变为 0，0 变为 1）。变异概率为 0.001 ～ 0.01。

7）循环

将新产生的子女样本加入原样本中，或可以直接加入前面被选出的优秀的父辈样本中，组成新种群。到此，一轮遗传操作完成。新种群返回到 3），再用每个样本对应的一组寻优参数计算其适配值，并按从优到劣的次序排列，并进行下一次迭代计算，直至达到满意的性能指标（或适配值）。在最后的种群中选择最好的一个样本，将其字符串解码后即得到最优的参数值。

由以上可以看出，遗传算法是模仿“优胜劣汰、适者生存”的生物进化过程，寻优参数的字符串编码类似于生物染色体中遗传基因的排列，对字符串的交叉和变异对应生物繁殖过程中遗传基因的重新组合和突变，而客观存在的生物进化法则保证了遗传算法的有效性和通用性。为了加快收敛速度而又保证得到全集最优解，对遗传算法亦有不少研究和改进。

14.4 遗传算法的实质

传统的遗传算法即可使用选择、交换和变异三种遗传操作，这三种非常简单的操作如何使遗传算法拥有如此强大的优化能力。Holland 在他所建立的模式理论中(Schemata Theory)首次从理论的角度回答了这个问题。

从模式的角度看，遗传算法的搜索过程其本质是对隐含在可行解编码串内的“模式”的抽样过程。关于遗传算法的收敛性，标准遗传算法在变异概率为 0 时，必然收敛于一个

吸收状态，但不能确保收敛于全局最优解；标准遗传算法在变异概率不为零时，是不收敛的。但遗传算法的一些变形形式，例如附加“记录已知最佳解”策略的遗传算法；采用最佳保留选择策略的遗传算法以及选用已知最佳个体构成种群的基本遗传算法均能在时间无限时以概率 1 收敛于全局最优解。

尽管已经证明了一些基本遗传算法最终能收敛到全局最优解，但所需时间可能是无限的。从实用角度看，在无限时间内收敛到全局最优解是没有实用价值的。所以一般地说，优化方法只能从较高的搜索效率和较好的优化效果之间进行权衡，以期获得综合收益。从这种意义上讲，提高在有限时间内搜索到全局最优解的概率，以及确定算法的时间复杂性，显然是更为重要的、具有实际价值的问题。

在遗传算法的时间复杂性上，首先，变异操作的存在增加了遗传算法的时间复杂性，其次，标准遗传算法的时间复杂性在变异概率较小的情况下主要与遗传算法的种群规模、染色体编码长度以及优化对象本身的可行解分布的特性密切相关。一般来说，遗传算法种群规模越大，染色体编码长度越长，则遗传算法时间复杂性越高。

遗传算法的进化机制包含着遗传变异（由交叉和变异操作提供）的产生和选择策略（有选择操作实现）两方面的综合作用，其选择操作是在适配度空间内进行的，而交叉、变异操作则是在基因型空间内进行，遗传算法的优化思想原本是期望通过有倾向性的选择，并借助交叉、变异操作的基因型重构能力，增加进入全局最优解所在区域的机会并搜索到全局的最优解。但由于遗传算法在适配度空间内的选择操作与在基因型空间的交叉、变异操作存在着明显的矛盾，使得这种优化思想尚不完善。总之，从全局最优化角度考虑，由于目前还难以准确评估进化意义上的个体适配度，这就使得基于个体适配度的交配选择存在相当的盲目性。

14.5 小　结

①遗传算法主要是靠种群基因型的多样性提供进化机会的，要使遗传算法产生不断进化的效果，就必须在整个优化过程中维持种群基因型的多样性。

②依据模式理论，遗传算法的搜索是对隐含在编码串内的模式抽样和编码串间的模式重构的过程。

③从机理上讲，遗传算法是依据种群内个体间的基因值和基因型的相似性来确定遗传算法的搜索方向的。

第 15 章 遗传算法的应用

15.1 采用遗传算法提高神经网络模型辨识的精度

15.1.1 引言

被控对象的模型辨识和系统的建模在控制系统的设计过程中占有重要的地位。即使是采用诸如模糊逻辑控制等这类不依赖于被控对象精确数学模型的控制策略，在设计、分析和系统仿真过程中，准确的系统输入/输出特性的获得，无疑使所设计的参数更加接近实际控制系统，在一定程度上，能够加快设计进程，提高设计质量，所以准确快速地获得复杂的被控对象输入/输出特性，在对控制的快速和高精度要求的今天，具有相当重要的意义。

离线批处理的系统参数辨识过程，只能辨识出线性系统或非线性系统在线性段上工作点的模型，从而限制了对非线性系统的应用，人工神经网络以其特有的自组织、自学习的能力，能够对无法通过物理公式推导出的复杂的或非线性系统的模型，进行输入/输出关系的特性训练，这一特性为非线性系统建模开辟了一条新的途径。不过，人工神经网络也存在着不足之处。首先，网络的结构设计没有统一的标准；其次，网络的学习训练是一个艰难的过程，离线训练往往需要很长时间，能否获得最优解，在很大程度上取决于初始值的随机性。为了获得最优化，常常需要重复多次训练。最大的不足是易于陷入局部极小值，而达不到学习的目的。

上述缺点是因为人工神经网络所采用的反向传播(BP)法本质上是梯度下降法，而 BP 网络的学习过程又是对一个高度非线性目标函数求全局最优解的过程，所采用的又是传统的一维搜索法。虽然人们已采用了各种改进措施试图解决上述问题，但这些措施也只能在某些特殊情况下有效，未能从根本上解决问题。

与人工神经网络依赖于起始点单点优化搜索不同，遗传算法(GA)保持了一个潜在解的群体，实现了多个方向上的搜索。同时，在搜索过程中鼓励不同方向上信息的结合和交流，使群体经历了一个模拟进化过程：每一代中，较好的解“再生”，较差的解“消亡”。从而大大减少了陷入局部解的可能性。遗传算法为解决人工神经网络所存在的问题，继而提高人工神经网络对系统特性辨识的精度和速度，提供了可能性。但对 GA 进行应用研究时，如何对所要解决的问题应用 GA，以及应用 GA 的成效，在一定程度上取决于设计者对遗传算法和所要解决问题的理解程度。在上一章里已经对遗传算法的理论和设计作了大体介绍，本节将采用遗传算法进行具体的优化建模网络的应用。为了能够加快收敛速度，对标准遗传算法进行适当的改进。

15.1.2 改进的遗传算法

1) 编码

应用标准的遗传算法进行变量的编码和解码，当变量众多、取值范围大或无法给定范围时，此种方法将使 GA 的收敛速度降低，同时在变量的编码与解码过程中，常会导致有用信息的丧失，而且存在精度和算法量之间的矛盾。所以我们对其进行改进：采用直观的实数编码，以每一权重值的每一自然数字作为基因，所有权重值的顺序排列组成一个染色体。这样既方便了后面所要进行的遗传操作，又具有直观性和实用的方便性。

2) 遗传操作

遗传操作主要包括复制、交换和突变，其目的是将当代群体变为下一代群体。其过程是：首先按正比于适配值的概率选择本代中的个体进行复制。通过复制，淘汰最差个体，但未创造出新的个体。交配操作能够造出新的个体。参加交配的两个个体也是基于适配值按比例选择的，交配结果产出两个新的个体。但当所有个体串全都一样时，交换操作则不能产生新的个体，这时由交配操作产生的后代的适配值不再比它们的前辈更好而又未达到全局最优解，这就发生了所谓的早熟收敛。早熟收敛的根源是发生了有效基因的缺失，为克服这种情况，只有依赖于突变。但常规的突变效果是不明显又很慢的。

在遗传操作过程中，适配值一直起着衡量操作结果优劣的作用。对于所要解决问题的不同，适配值表达式的内容也是不同的。在系统输入/输出特性辨识的问题中，适应值一般采用样本输出与神经网络输出信号之间差的平方和，其目标是求得适配值最小时的一组神经网络的权重值。

为了保持解的多样性，并防止早熟现象，在辨识过程中，对遗传操作进行了如下改进：

在复制过程中，在 N 个个体中将小于平均适配值的个体复制下来，淘汰高于平均适配值的劣值个体 n 个，被淘汰的 n 个个体的空缺，采用随机取数方式进行增补，当 n 小于$(1/4)N$ 时，随机增补的个体数取为$(1/4)N$。保持$(1/4)N$ 的随机个体数，是为了保持群体交换过程中的多样性。

为了确保搜索的全局最优，将本代中的最优解直接进化到下一代中。代表权重值的实数取为一位整数和四位小数的数字，以保持足够的精确度。在交换操作中，将上一代最优值放入第一个个体，将 2 到 $N/2$ 之间的个体的后三位小数与 $N-1$ 到$(N/2)+1$ 之间的个体数值中的后三位小数进行相互交换，形成新的$(N/2)-1$ 个个体，同时，对于$(N/2)+1$ 到 N 的个体的选取，用其后三位小数与 1 到 $N/2$ 的个体后三位小数之间进行交换来获得。在交换后所获得新的 N 个个体中，从$(N/4)+1$ 到 $3N/4$ 个个体是经过次优个体之间的交换得到的，而前 $N/4$ 和后 $N/4$ 的个体，因为与其交换的个体中均有 $N/4$ 个个体为随机数，所以它们与次优个体交换后，可以产生新的个体，尤其在经过数代进化后，这些不断变化的新个体有效地延缓了早熟的出现。在交换位数的选择上，实验证明后三位小数的同时交换的效果最好，优化效率最高。取一位或二位时的进化收敛速度都较慢。

由于搜索空间的性质和随机值的优劣，在进化过程中，仅靠交换往往会出现早期陷入局部解而中止进化。配合有效的变异操作可防止其出现。在所应用的实数编码中，采用的变异法是将某个个体中的某一自然数变为另一自然数，而不是按标准 GA 中改变某一位置

上的数字。其目的是为了增加随机性，进而增加多样性。随着进化代数的递增，进行突变次数的比例数也随之增加，以便在接近最优解的群体中能够增加更多的新的个体。突变比例次数在整个搜索过程中是由 0.05 变化到 0.12。

15.1.3 实例验证

为了验证所提出的改进遗传算法的有效性，将其用于对具有严重非线性摩擦力影响的直流电机速度的输入/输出特性的辨识中。为了获得有效的动态性能的辨识，人工神经网络的结构采用双输入、单输出，双输入分别为力矩以及输出的一阶延迟信号。网络节点结构取为（2 - 5 - 1），隐含层中采用 S 型函数，输出层为线性函数。网络的结构如图 15.1 所示。选取群体数 $N = 50$，在最大进化代数 200 完成以后，误差平方和达到 0.0042，其辨识后的验证效果如图 15.2 所示，其中实线为对象的输出，虚线为网络模型的输出。

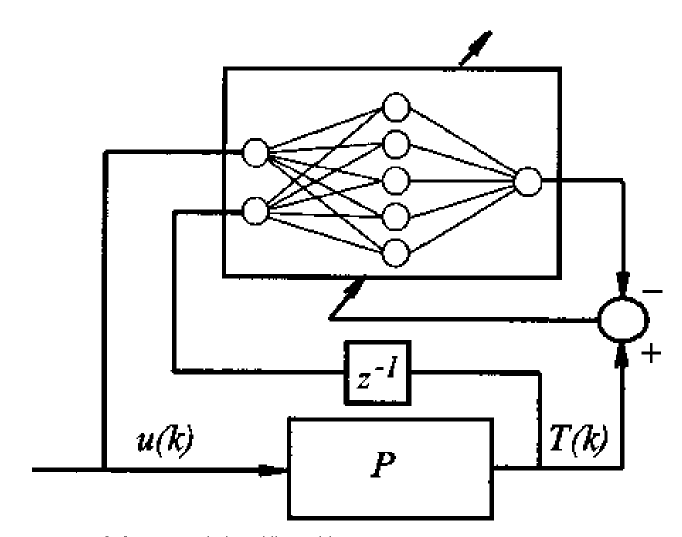


图 15.1 用于辨识的人工神经网络的结构图

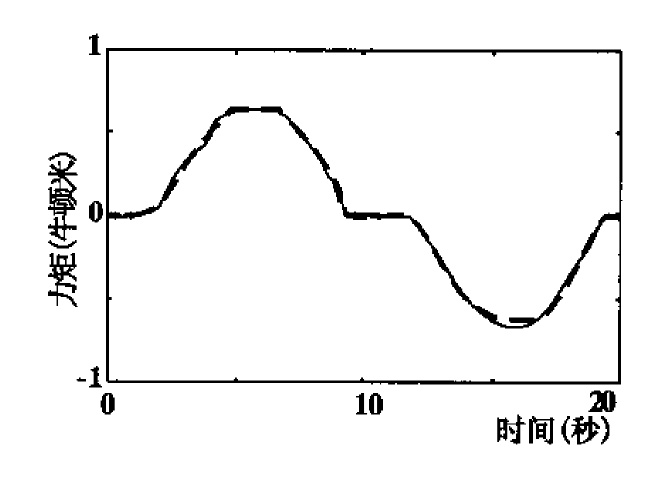


图 15.2 辨识后的网络输出与实际对象输出

作为对比，对同一对象采用带有自适应学习速率的反向传播(BP)法进行辨识。在经过2000次循环训练后，所达误差平方和为0.0088。两种方法的结果如表15.1所示。从中可以看出，采用改进的遗传算法对加速收敛速度、提高辨识精度有着明显的效果。

表15.1 两种寻优方法的对比结果

算　法	训练次数	达到最小误差和
自适应BP	2000	0.0088
改进GA	200（代）	0.0042

15.1.4 小结

改进的遗传算法可以有效地加速优化收敛速度，提高对系统辨识的精度，尤其在对复杂系统进行模型辨识时，更能体现出它的优越性。

15.2 模糊神经网络和遗传算法相结合的控制策略

15.2.1 引言

近年来，模糊逻辑控制、神经网络等技术得到了迅速发展，并引起了人们的广泛关注。由于模糊逻辑控制具有较强的鲁棒性和灵活性，因而其在一些非线性、动态特性变化大和无法进行数学建模的系统中显示出明显的优越性。然而由于模糊逻辑控制从本质上来说是基于人的经验，故在选取合适的对控制效果起决定性作用的隶属函数和控制规则时，先验知识就显得尤为重要。但是这种先验知识一般来说往往是比较缺乏或不全面的，特别对某些复杂的和非线性系统来说，根本就不可能得到详细或准确的先验知识，这就为模糊逻辑控制的有效实施和精度的提高带来了一定的困难。

为了解决这一困难，人们一方面不得不采用较多的模糊子集以及更多的控制规则；另一方面也在不断地研究可以自动生成、修改以及优化隶属函数和控制规则的方法与技术。然而在目前尚无成熟调试方法的情况下，仅凭经验要想对众多的待调参数得到一合适的组合是相当困难的，另外，正是因为存在诸多的待调参数，使得人们往往只能针对模糊逻辑控制的某一方面进行优化，例如仅对比例因子进行在线修改以改善系统性能，或仅对隶属函数的形状和位置进行优化，或只是修整模糊控制规则等等。但是，模糊逻辑控制的总体效果是所有模糊化、模糊推理和解模糊等诸多方面的有机综合，只有对各部分同时进行的优化才可望达到最优的控制效果。另外，这些研究对如何将模糊子集和模糊规则的个数减少到最少，同时又不降低模糊逻辑控制的效果涉及较少。模糊子集和模糊规则数的多寡是体现模糊逻辑的复杂程度的重要标志以及影响模糊控制物理实现的重要因素。

本节给出了一种将模糊逻辑控制、神经网络和遗传算法有机结合而进行全局优化的控

制策略，为解决所存在的困难提供了可能的途径。首先，采用人工神经网络实现模糊逻辑控制的全过程，从而使神经网络的结构具有模糊逻辑控制的功能，同时每一个参数均具有明确的模糊逻辑意义；其次，利用人工神经网络对信息处理具有自学习和自适应的特性对其参数和结构同时进行优化处理，而达到在最简结构下获得最优控制效果的目的。为了加快学习速度以及避免陷入局部极小值，文中引进了遗传算法。所提控制策略的实质是将多种方法和技术进行有机的综合，使它们之间取长补短，在最简的结构下优化全部参数，达到最佳控制效果。本章节在给出优化控制系统结构后，对优化后的仿真结果进行了对比分析，并且给出了优化后的隶属函数图以及控制器的输入/输出特性曲面图。

15.2.2 优化控制系统的结构

优化控制系统的结构如图 15.3 所示，其中 P 为被控对象；NNP 为采用被控对象的输入/输出信号，通过训练而辨识出的 P 的神经网络模型；GNFC 为基于遗传算法优化的具有明确模糊逻辑意义和模糊推理功能的神经网络控制器。控制器的训练是通过使 GNFC 与 NNP 相串联，连接成负反馈控制回路，并采用遗传算法，使目标函数 $J = \min\sum[y_d(k) - y(k)]^2$ 来求得最优的代表隶属函数和控制规则的权重值。

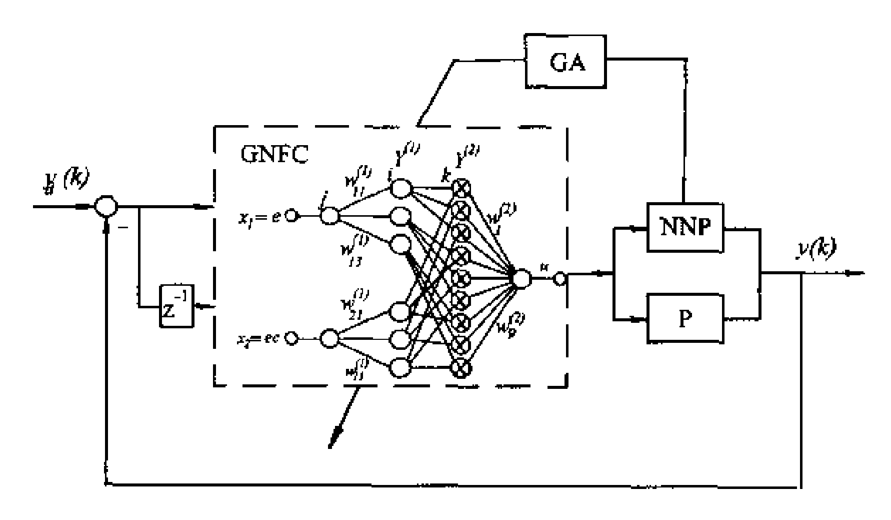


图 15.3 优化控制系统的结构图

1）具有模糊逻辑功能的神经网络控制器 GNFC

具有模糊逻辑功能的神经网络控制器 GNFC 的结构如图 15.3 中虚框所示。可以看出它是一个具有输入层、中间层和输出层的三层神经网络。该神经网络与普通神经网络是有很大不同的。在结构上，首先在于输入层的非全联结，从图中可以看出，x_1 和 x_2 各自对应三个节点；其次在于输入层的标准正态转移函数和输出层的线性转移函数；最后在于中间层是乘法器而非严格的神经网络节点。该层的功能是将来自输入层的两组各三个信息的数据两两相乘后所得的 9 个乘积值送到输出层。

功能上，该网络的三层节点是严格对应于模糊逻辑控制的模糊化、规则推理和逆模糊三个步骤的，因而具有明确的模糊逻辑意义。首先，网络输入量为 e 和 ec；其次，输入

层节点的转移函数代表的恰为模糊变量的隶属函数。该层的权重值 $W_{ij}^{(1)}$ 和 θ_{ij} 的不同也就意味着变化多端的隶属函数的形状和位置。该层的输出 $Y^{(1)}$代表的就是模糊化的结果—隶属度。再次，中间层是将模糊化得到的隶属度两两相乘的功能。中间层的输出代表着 9 个模糊规则的规则强度，将这些强度传递给下一层就可以进行解模糊了。最后，输出层的各个权重值 $W^{(2)}$代表了九条规则，根据重心法的解模糊方式，只要将它们作为权重值与输入，即规则强度加权求和，输出即为模糊控制的输出量。此处采用的模糊推理方式为代数积－加法－重心法。尽管这种推理方式的模糊逻辑的意义相对极小－极大推理方式较小，但是它可以降低推理过程中控制信息的损失，同时又更容易用神经网络来实现。

总的来说，GNFC 的输入/输出关系可以归结如下：

$$X=[e \quad ec]^{\mathrm{T}} \tag{15.1}$$

$$y_{ij}^{(1)}=\exp[-(W_{ij}^{(1)}\cdot x_j+\theta_{ij})^2] \tag{15.2}$$

$$y_k^{(2)}=y_{m1}^{(1)}\cdot y_{n2}^{(1)}=\exp\{-[(W_{m1}^{(1)}\cdot x_1+\theta_{m1})^2+(W_{n2}^{(1)}\cdot x_2+\theta_{n2})^2]\} \tag{15.3}$$

$$u=W^{(2)}\cdot Y^{(2)}=\sum_{k=1}^{9}y_k^{(2)}\cdot W_k^{(2)} \tag{15.4}$$

其中 $j=1,2$；$i,m,n=1,2,3$；$k=1,2,\ldots,9$。

如果脱离具体的神经网络结构，而直接采用乘法－加法－质心法的推理方式和正态隶属函数，同样可以得到如上公式。之所以如此，是因为经过这样的从模糊逻辑到神经网络结构的映射和结合，即可将模糊逻辑的最重要的参数—隶属函数的形状和位置，以及模糊规则转化成神经网络的权重值，这样就可以利用神经网络的自学习、自适应能力来修正和优化它们，也就可以达到优化模糊逻辑控制效果的目的了。另外，对于该系统的输入输出我们作了归一化的处理，这样可省去待调的三个比例因子，而将它们的作用和调整直接纳入到了神经网络参数中，以此方式该网络可以优化所有影响模糊逻辑控制效果的参数值，因而也就为全局最优奠定了基础。此时，对模糊控制器参数的优化就转化为对神经网络 21 个可调整参数的优化。因模糊子集数为 3 个，模糊规则为 9 个，就模糊逻辑控制的物理实现来说已达到最少。

2) 用于过程模型辨识 NNP

图 15.3 中的用于过程模型辨识 NNP 是用来辨识被控对象的输入/输出特性，以获得进行控制器性能优化所需的被控对象的模型。由于对控制器参数的优化过程采用的是间接法，即对由被控对象和控制器串联并经过负反馈所组成的闭环回路，采用期望输出与实际系统输出之差的平方和作为目标函数，以此来确保全局优化质量，这样，被控对象的输入/输出特性的辨识就自然成为控制器设计的一部分。此法避免了由直接采用控制器输入/输出数据进行训练优化所带来的非最优函数的逼近问题。在模型辨识上为了获得动态特性，采用含有一个隐含层且带有输出一阶延迟的回馈的前向网络，网络内部结构分别采用 S 型和线性转移函数，并通过实验选取最少的五个隐含节点。网络的训练采用具有自适应学

习速率的 BP 算法，训练后的网络在一定的误差范围内具有实际被控过程的输入/输出特性。

对于 GNFC 的训练可以采用 BP 算法，但是由于其中间层是一乘法器，故必须对 BP 算法作进行修正后方可使用，而修正后的 BP 算法不仅收敛速度慢，而且很容易陷入局部极小值，因此，决定采用遗传算法（GA）进行优化。

3）*改进的寻优遗传算法*

首先将所辨识的过程模型与待求的神经网络控制器相串联，并形成负反馈控制回路，再利用改进的遗传算法进行运行仿真，以寻求得到模糊控制中的最优化隶属函数和控制规则的组合控制效果。

编码：首先定义包含所求变量的个体。为了操作方便以及精度的需要，采用实数编码，直接将待处理的权重值逐位数字地顺序排列，并转化成数字字符串，形成解的个体。由 N 个个体形成种群。

适配值：适配值函数 F_i 采用系统的期望输出与实际系统的输出之差的平方和来定义，即：

$$F_i = \sum_{k=1}^{M} E_i^2(k) = \sum_{k=1}^{M}\left[y_d(k) - y_i(k)\right]^2 \tag{15.5}$$

其中，$i = 1, 2, \ldots,\ N$ 为种群中的个体数，k 为个体中待求变量数。优化的目的是使适应度 F_i 达到某个满意的指标。

遗传操作：在复制操作中，淘汰 $0.25N$ 个低于平均适配值的劣解，并以随机取数的方式补齐，以保持群体交换过程中解的多样性。为了确保搜索的全局最优，在进行交换操作前，首先将本代样本中的最优解直接进化到下一代中。除此之外，每个个体均按一定的比例两两进行多位或一位相互交换，以形成新的个体。复制中新增补的随机个体，尤其在经过数代进化后，与次优个体交换后不断变化出的新个体，可以有效地延缓早熟的出现。

当进化接近最优解时，仅由交换操作产生的后代的适配值可能不再比它们的前辈更好，此时将某一位置的自然数变为另一自然数(不同于标准 GA 中改变某一位置上的数字)，目的是为了增加随机性，进而增加多样性。另外，随着进化代数的增加，进行突变的比例也随之增加，以便在接近最优解时的群体中增加更多的新个体。突变比例在整个搜索过程中由 0.05 变化到 0.12。

下面的例子可以很好地说明交换操作和变异操作的过程。例如：个体中两个变量分别为 0.5712 和-1.2327，当它们以一定方式被选中后，取后三位小数进行交换，形成新的数 0.5327 和 -1.2712，然后通过变异操作又将其中的某一随机数，如“2”，即 0.5327 和 -1.2712，变异成“3”，则形成新一代个体中的变量：0.5337 和 -1.3713，变异操作在越接近最优值、交换操作趋于一致而不再有变化时，越能体现出它的重要性。

通过以上处理，控制系统的参数优化过程可以归纳如下：

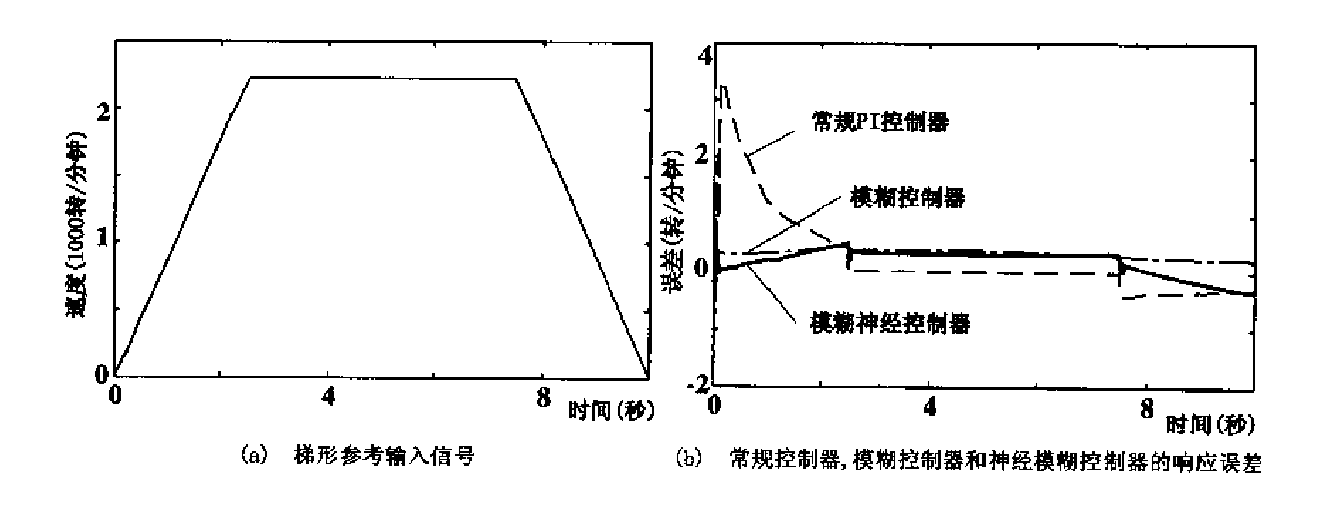
①训练得到被控对象的网络模型 NNP；

②根据模糊逻辑控制的最少模糊子集及控制规则构造神经网络控制器 GNFC；

③将 NNP 与 GNFC 联成闭环回路，并采用改进的遗传算法进行参数优化。

15.2.3 优化仿真结果的对比和分析

用于控制策略验证的仿真系统为一具有严重非线性摩擦力的直流电机速度控制系统。输入参考信号为一梯形输入。我们将 GNFC 控制器的跟踪效果与常规 PI 控制器以及具有 5 个论域、采用三角形交叉隶属函数，以 $u = (e+ec) / 2$ 为控制规则的模糊逻辑控制器的跟踪效果进行了比较，图 15.4 即为三者的跟踪响应，其中，图 15.4(b)为误差响应。从图中可以看出，GNFC 在零速度附近表现出比常规控制器和普通的模糊逻辑控制器均佳的由非线性摩擦力所造成的难以克服的跟踪误差，而且几乎达到了稳态误差为零的完美程度。这表明它可以同时兼顾到非线性特性和稳态精度两方面的要求。



(a) 梯形参考输入信号　　(b) 常规控制器，模糊控制器和神经模糊控制器的响应误差

图 15.4 控制系统仿真结果对比图

为进一步分析优化后的控制器性能，下面我们来直观地看一看优化后的模糊逻辑的隶属函数图及控制器输入/输出特性曲面图，它们分别如图 15.5 和图 15.6 所示。从图 15.5 中可以看出优化后的隶属函数图与常规的一些人为设定的隶属函数图是很不相同的，它没有均匀的分布性，这也正体现了优化系统针对具体被控对象优化得到具有具体特点的隶属函数的功能。e 和 ec 的隶属函数是不同的，并且各自都有特色。另外，每一模糊子集的隶属函数的伸展宽度是不一样的：有的较窄，有的相当宽，几乎在一定区间内成了直线。再次，这些函数的中心位置不是等距的，这体现了优化结果的丰富性。

图 15.6 描绘的是优化后 GNFC 的输入 e 和 ec 与输出 u 的关系图，这种描述更直观、形象。本节所讨论的被控对象是具有严重非线性的，这一点恰在该图中被形象地体现出来了，特性曲面图中的“峰”与“谷”的变化代表的正是系统所固有的非线性。从这一点来说，优化系统不需要先验的经验知识，它可以通过训练自动地找出适应系统的控制特性来，这一功能无疑是具有重要意义的。从图中还可以看出，在 $e = ec = 0$ 时，图 15.6 中的 u 为一正的数值。我们知道，速度系统的非线性摩擦力始终是存在的，故即使误差为 0 且不再变化，为了维持这一现状，控制量为 0 是不行的。GNFC 使 u 不为 0，符合实际的情况，因而其稳态误差自然要比普通模糊控制器要小得多。另外，对于控制量在零速度附近也有

较大的差异，这可以从它们的特性曲线图中得到合理的解释。

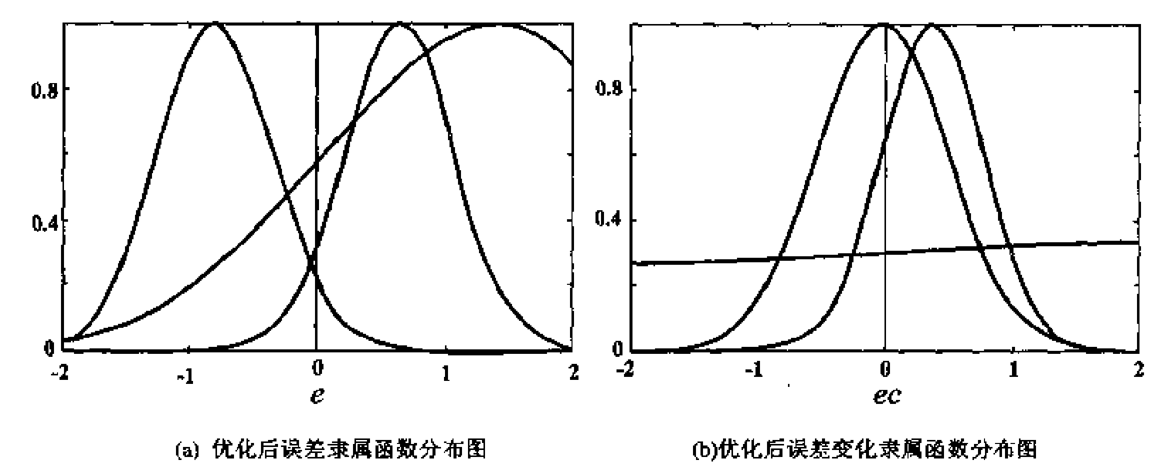


图 15.5 优化后的模糊逻辑的隶属函数图

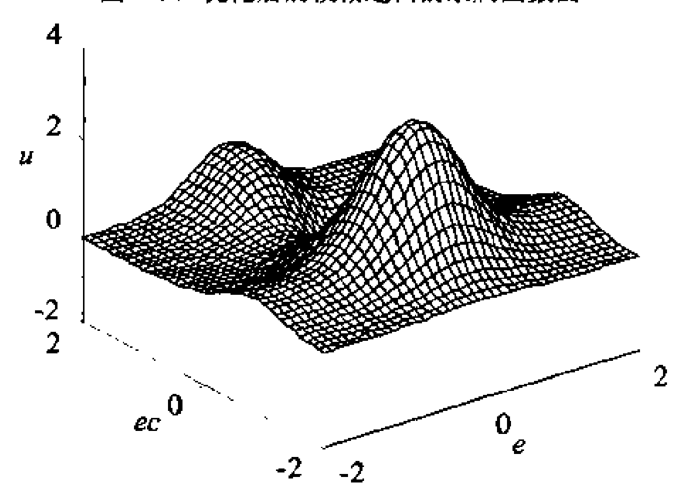


图 15.6 输入 e 和 ec 与输出 u 的关系图

15.2.4 小结

本节给出并讨论了一种模糊神经网络和遗传算法相结合的控制策略，通过设计一种具有模糊逻辑意义的神经网络控制器，并且结合遗传算法的全局寻优特性，完成了对模糊逻辑的隶属函数的形状位置和模糊规则的最优化，并且在数量上将它们降到最低。针对非线性系统所作的优化在仿真对比中表现出了较常规控制器和普通模糊控制器好的多的控制效果。通过观察和对比模糊逻辑的隶属函数和输入/输出特性曲面图，可直观地了解这种优化控制策略的真正内涵，即它可以自动提取复杂系统的内在特性，通过优化将它反应到模糊逻辑的特性之中，以形成一个针对该复杂系统的最优控制策略。

第 16 章 模拟退火算法及其应用

不论是神经网络的权值训练，还是模糊神经网络的参数优化，不论是采用基于梯度下降法还是数值优化方法进行求解，都存在局部极小值的问题。第 14 章所讲到的遗传算法是一种全局优化算法。本章所谈的模拟退火算法(Simulated Annealing Algorithm，简称 SAA)也是一种适合解大规模组合优化问题，特别是解 NP 完全问题的通用有效的全局优化解法，它与其他近似算法相比，具有描述简单、使用灵活、运用广泛、运行效率高和较小受初始条件限制等优点，而且特别适合并行运算。

模拟退火算法是将组合优化问题与统计力学中的热平衡问题类比，找到求解优化问题的新方法。对固体问题进行退火时，通常先将它加温熔化，使其中的粒子可以自由运动，然后随着温度的逐渐下降，粒子也逐渐形成了低能态的晶格。若在凝固点附近的温度下降足够慢，则固体物质一定会形成最低能态的基态。对于组合优化问题来说，也有这样的类似过程。1982 年，Kirkpatrick 等将退火思想引入组合优化领域，提出一种解大规模组合优化问题，特别是 NP 完全组合优化问题的有效近似算法——模拟退火算法。它通过对固体退火过程的模拟，采用 Metropolis 接受准则，并用一组成为冷却进度表的参数控制算法进程，使算法在多项式时间里给出一个近似最优解。

16.1 Metropolis 准则和模拟退火算法

固体在恒定温度下达到热平衡的过程可以用 Monte Carlo 方法进行模拟。Monte Carlo 方法的特点是算法简单，但必须大量采样才能得到比较精确的结果，因而计算量很大。从物理系统倾向于能量较低的状态，而热运动又妨碍它准确落入最低态的原理出发，采样时着重取那些有重要贡献的状态，则能够较快地达到较好的结果。

1953 年，Metropolis 等提出了重要性采样法。他们用下述方法产生固体的状态序列：先给定以粒子相对位置表征的初始状态 i，作为固体的当前状态，该状态的能量是 E_i，然后用摄动装置使随机选取的某个粒子的位移随机地产生一个微小的变化，得到一个新状态 j，新状态的能量是 E_j，如果 $E_j < E_i$，则该新状态就作为“重要”状态，如果 $E_j > E_i$，则考虑到热运动的影响，则该新状态是否可作为“重要”状态，要依据固体处于该状态的几率来判断。而固体处于状态 i 和 j 的几率的比值等于相应 Boltzmann 因子的比值 P，即：

$$P = \exp(\frac{E_i - E_j}{kT}) \tag{16.1}$$

其中，P 是一个小于 1 的数。

用随机数发生器产生一个[0,1]区间的随机数 random，若 $P >$ random，则新状态 j 作为重要状态，否则舍去。若新状态 j 是重要状态，就以 j 取代 i 成为当前状态，否则仍以 i 为当前状态。重复以上新状态的产生过程，在经过大量固体状态的如此变换后，系统逐渐趋于能量较低的平衡状态，固体状态的概率分布趋于指数形式的正则分布。

上述接受新状态的准则称为 Metropolis 准则，相应的算法称为 Metropolis 算法。

1982 年，Kirkpatrick 等首先意识到固体退火过程与组合优化问题之间存在的类似性，加上 Metropolis 等对固体在恒定温度下达到热平衡过程的模拟也给他们以启迪：应该把 Metropolis 准则引入到优化过程中来。最终他们得到一种对 Metropolis 算法进行迭代的组合优化算法，这种算法模拟固体退火过程，称之为"模拟退火算法"。

模拟退火算法的求解过程如下：

设组合优化问题的一个解 i 及其一个目标函数 $f(i)$分别与固体的一个微观状态 i 及其能量 E_i 等价，令随算法进程递减其值的控制参数 t 担当固体退火过程中的温度 T 的角色，则对于控制参数 t 的每一个取值，算法持续进行"产生新解—判断—接受/舍取"的迭代过程就对应着固体在某一恒定温度下趋于热平衡的过程，也就是执行了一次 Metropolis 算法。与 Metropolis 算法从某一初始状态出发，通过计算系统的时间演化过程，求出系统达到的状态相似，模拟退火算法从某个初始解出发，经过大量解的变换后，可以求得给出控制参数值时组合优化问题的相对最优解。然后减小控制参数 t 的值，重复执行 Metropolis 算法，就可以在控制参数 t 趋于零时，最终求得组合优化问题的整体最优解。由于固体退火必须"徐徐"降温，才能使固体在每一温度下都达到热平衡，最终趋于能量最小的基态，控制参数的值也必须缓慢衰减，才能确保模拟退火算法最终趋于组合优化问题的整体最优解。理论上已经证明：模拟退火算法以 1 的概率收敛到整体最优解集，但渐近收敛到最优解集须经历无数次变换。模拟退火算法的渐近收敛性意味着：对于多数组合优化问题来说，它是一种指数型时间算法，算法的执行过程只有进行无限多次变换后，才能返回一个整体最优解。为了能够在有限时间里获得一个较优的解，人们引入了冷却进度表(cooling schedule)。冷却进度表是一组控制算法进程的参数，用以逼近模拟退火算法的渐近收敛性态，使算法在有限时执行过程后返回一个近似最优解。因而冷却进度表是影响模拟退火算法实验性能的重要因素，其合理的选取是算法应用成败的关键。

16.2 模拟退火算法的设计步骤

模拟退火算法应用的一般步骤是：从选定的初始解开始，在借助于控制参数 t 不断递减的过程中产生的一系列 Mapkob 链中，利用一个新解产生装置和接受准则，重复进行包括"产生新解—计算目标函数差—判断是否接受新解—接受（或舍取）新解"这四个过程，不断对当前解进行迭代，从而达到使目标函数最优的目标。因此，算法的应用需满足以下三个方面的要求。

1）*数学模型：由解空间、目标函数和初始解三部分组成*

解空间：它为问题的所有可能（可行的或不可行的）解的集合，它限定了初始解选取和新解产生的范围。在许多组合优化问题中，一个解除了满足目标最优的要求外，还必须满足一组约束，因此在解集中往往可能包含一些不可行解，为此，可以限定解空间仅为所有可行解的集合，即在构造解时就考虑到对解的约束，这可以通过在目标函数中加上所谓的惩罚函数以"惩罚" 不可行解的出现。

目标函数：组合优化问题的目标是从组合问题的可行解集中求出最优解。优化问题有

三个基本要素：变量、约束和目标函数。在求解过程中选定的基本参数称为变量，对变量取值的种种限制称为约束，表示可行方案衡量标准的函数称为目标函数。目标函数是对问题的优化目标的数学描述 f，通常表述为若干优化目标的一个和式。目标函数的选取必须正确体现对问题的整体优化要求。例如，如上所述，当解空间包含不可行解时，目标函数中应包含对不可行解的惩罚函数项，借此将一个有约束的优化问题转化为无约束的优化问题。一般地说，目标函数常常是一个误差函数的平方和，不过，目标函数也不一定就是问题的优化目标，但其对应关系应是明显的。此外，目标函数式应当是易于计算的。

初始解：它是算法开始迭代的起点。初始解的选取应当使得算法导出较好的最终解。不过大量实验表明，模拟退火算法的最终解并不十分依赖初始解的选取。

2）新解的产生和接受机制

新解的产生和接受可分为以下四个步骤：

首先，由一个产生装置从当前解空间中产生一个新解。为便于下面的计算和接受（这正是算法中最耗时的工作），通常选择由当前解经过简单变换即可产生新解的方法，如对构成解的全部或部分元素进行置换、互换等。

其次，计算与新解伴随的目标函数的差。

第三，判断新解是否被接受。判断的依据是一个接受准则。最常用的接受准则是 Metropolis 准则，即：

$$r=\begin{cases}1, & \Delta f<0\\ \exp(-\Delta f/t), & \Delta f\geq 0\end{cases} \tag{16.2}$$

其中，t 为控制参数，Δf 为新解与当前解之间的目标函数差。

最后，当新解被确定接受时，用新解代替当前解。此时，当前解实现了一次迭代。可在此基础上开始下一次新的迭代。如果所获新解被判定为舍弃时，那末在原当前解的基础上继续进行下一次新的迭代。

3）冷却进度表

它是用于控制算法进程的一组参数的集合，包括控制参数的初始值及其衰减函数、对应的 Mapkob 链的长度和停止准则。冷却进度表的选取是算法性能的主要因素。

下面将结合采用模拟退火算法求解 TSP 问题的具体实例，详细解释应用模拟退火算法解决实际问题时必须考虑的方面。

16.3 应用模拟退火算法求解 TSP 问题

模拟退火算法实际上是一种随机搜索算法，即它在一定的解空间内随机地产生新解并对其进行取舍。所不同的是，对一个求目标函数最小值的问题，一般的随机搜索算法根据新解对应的目标函数值是否减小来判断是否接受新解；而模拟退火算法则允许接受使目标函数增大的新解，其优点在于能跳出局部极小值，有利于找到全局最小值。

模拟退火算法的核心是在获得新解 P_{ij} 后，根据一定的准则产生新解的接受概率 P_{ij}，如取 Metropolis 准则，新解的接受概率如(16.2)式所示，可重写如下为：

$$r_{ij}=\begin{cases}1, & f_j \le f_i \\ \exp(-\frac{f_j-f_i}{t}), & f_j > f_i\end{cases} \tag{16.3}$$

其中，i 是已知解，j 是新产生的解。

当$f_j>f_i$时，还要进行如下判断：随机产生一个服从(0,1)的随机变量 random，若 $r_{ij} \geqslant$ random 则接受j，否则不接受。这样处理的好处是：倘若我们将满足$f_j \leqslant f_i$的解j称为好解，将满足$f_j > f_i$的解称为坏解，那么采用 Metropolis 准则，可以使我们在寻找好解的过程中允许接受坏解，从而能在一定程度上避免陷入局部极小值。

在模拟退火算法的求解过程中，t 是一个不断减小的控制参数，随着 t 的减小，坏解被接受的概率就越小。当 t 足够小之后，模拟退火算法就变成了纯粹的随机搜索算法，从而保证了求解过程的收敛性。倘若将产生并接受一个新解称作一次变换，模拟退火算法可以简单地描述为：在控制参数一定的情况下，经过无限次的变换，算法得到一个稳态值；随着控制参数从一个较大的值逐渐变为 0，这些稳态值最终将收敛于问题的最优解。因此，从理论上说，模拟退火算法是一种最优化算法，但是实际上，模拟退火算法并不能在多项式时间内求得优化问题的最优解。这并不意味着模拟退火算法没有实际意义，根据模拟退火算法的原理，我们可以在多项式时间里对模拟退火算法的渐进性态进行逼近，从而得到问题的近似最优解。

16.3.1 TSP 问题的求解步骤

现以旅行商(TSP)问题为例来说明在实现模拟退火算法的过程中需要考虑的问题。

设有 n 个城市和距离矩阵 $D=[d_{ij}]$，其中，d_{ij} 表示城市 i 到城市 j 的距离，$i,j=1,2,\cdots,n$。问题是要找遍访每个城市恰好一次的一条回路，且其路径长度为最短。

采用模拟退火算法求解的描述如下：

1) 解空间

用模拟退火算法求解 TSP 的解空间 S 可表示为$\{1,2,\cdots,n\}$的所有循环排列的集合，即

$$S=\{(\pi_1,\pi_2,\ldots,\pi_n) \mid (\pi_1,\pi_2,\ldots,\pi_n)\text{为}\{1,\ldots,\ n\}\text{的循环排列}\} \tag{16.4}$$

其中，每一循环排列表示遍访 n 个城市的一条回路，$\pi_i=j$ 表示在第 i 个次序访问城市 j。

2) 目标函数

对于此问题的目标函数可选为：访问所有城市的路径长度，即

$$f(\pi_1,\cdots,\pi_n)=\sum_{i+1}^{n} d_{\pi_i\pi_{i+1}} \tag{16.5}$$

目的是求上述目标函数的最小值。解中约定$\pi_{n+1}=\pi_1$。初始解可选为$(1,2,\cdots,n)$。

3) 新解的产生

在叙述新解产生的方法之前，我们先定义一个 TSP 的 k 变换领域如下：

定义 16.1 （TSP 的 k 变换领域$(k\geq 2)$） 设 N_k 是一个 k 变换领域结构，则 k 变换领域定义为：$N_k(i)=\{j\in S \mid j$ 可由i经一次k变换得到$\}$，其中 i 和 j 分别是给定的实例的两个不同的解。

所谓 k 变换是将 i 对应的路径去掉 k 条边，然后用另外 k 条边放入，以得到 j 对应的路径。

新解可通过分别或交替使用以下两种方法来产生。

(1) 2 变换法

任选两个城市的访问序号 u 和 v(设 $u<v$)，逆转 n 和 v 之间的访问顺序，得到：

$$\pi_1,\ldots,\pi_{u-1},\pi_v,\pi_{v-1},\ldots,\pi_{u+1},\pi_u,\pi_{v+1},\ldots,\pi_n$$

2 变换的邻域规模是：$\Theta=\frac{1}{2}(n-1)(n-2)$。

(2) 3 变换法

任选序号 u，v 和 w（$u\le v<w$），将 u 和 v 之间的路径插到 w 之后访问，得到新的路径为：

$$\pi_1,\ldots,\pi_{u-1},\pi_{v+1},\pi_{v+2},\ldots,\pi_w,\pi_u,\pi_v,\pi_{w+1},\ldots,\pi_n$$

3 变换的邻域规模是：$\Theta=\frac{1}{2}(n-1)(n-2)(n-3)$。

不论用 2 变换法还是用 3 变换法，新解的产生概率都服从均匀分布。

4) 目标函数差

由（16.5）式已知 TSP 问题的目标函数的定义。为避免重复计算，同时提高计算效率，不需要计算每个新产生的解所对应的目标函数值。计算两个解所对应的目标函数值的差更容易。下面根据不同的情况来讨论新解与旧解的目标函数差（设 I 为旧解，j 为新解）。

(1)对于用 2 变换法产生新解的情况

$$\Delta f=f_j-f_i=(d_{\pi_{u-1}\pi_v}+d_{\pi_u\pi_{v+1}})-(d_{\pi_{u-1}\pi_u}+d_{\pi_v\pi_{v+1}})$$

(2) 对于用 3 变换法产生新解的情况

$$\Delta f=f_j-f_i=(d_{\pi_{u-1}\pi_{v+1}}+d_{\pi_w\pi_u}+d_{\pi_v\pi_{w+1}})-(d_{\pi_{u-1}\pi_u}+d_{\pi_v\pi_{v+1}}+d_{\pi_w\pi_{w+1}})$$

实际上，对于 3 变换法产生新解的情况，由于计算 Δf 所需判断的条件太多，效率未必比直接计算新解的目标函数值高，所以在本例应用时是直接计算新解对应的目标函数的值。

5) 新解接受的判断条件

接受准则有很多种，所采用的是 Metropolis 准则。设 P_{ij} 表示由旧解 I 产生新解 j，解 j 的接受概率值由（16.3）式决定。

16.3.2 冷却进度表的选取

模拟退火算法渐进收敛性的有限时间逼近通常采用以下方法来实现：用控制参数 t 的一个递减有限序列 $\{t_k\}$，$k=0,1,2,\ldots$，以及与之对应的链长为 L_k 的有限长齐次 Mapkob

链序列去控制算法进程。算法中涉及到的一些参数总称为冷却进度表，它包括如下参数：

①控制参数 t 的初值 t_0；

②控制参数 t 的衰减函数；

③控制参数 t 的终值(停止准则)；

④Mapkob 链的长度 L_k。

显然，冷却进度表的选取对算法而言是至关重要的。下面就上述几个参数的选取一一进行讨论。

1) 控制参数 t 的初值 t_0

控制参数 t 的初值 t_0 的选取原则是使算法开始时接受坏解的概率尽量大，这就要求 t_0 应该选得足够大。使最初的扰动在此温度时能够充分接受，同时使系统处于最无序状态，以模拟物质在高温时的混乱状态。但也不能无节制地取得太大，这样会导致算法在高温时停留过长，增加了算法的执行时间。

已有不少选取 t_0 的方法，下面采用 Aarts 等人提出的方法：

先对控制参数的某个确定的值产生一个 m 次尝试的序列，记 m_1 和 m_2 分别为目标函数减小和增大的变换数，Δf^+ 是 m_2 次目标函数增大变换的平均增量。设定平均接受概率值 χ，则 χ 满足：

$$\chi = \frac{m_1 + m_2 \cdot \exp(-\Delta f^+ / t)}{m_1 + m_2} \tag{16.6}$$

这样，t_0 就可以求得：

$$t_0 = \frac{\Delta f^+}{\ln \dfrac{m_2}{m_2 \chi - m_1(1-\chi)}} \tag{16.7}$$

只要将 χ 设置为初始接受率，就能求出相应的 t_0 值。初始接受率常取值为 0.9。

2) 控制参数 t 的衰减函数

选择一个好的冷却速率的控制参数 t 也是很重要的。由于降温过程中，系统能量的下降是平稳的，因而在系统能量下降剧烈的地方，应适当放慢冷却速度，使系统达到平衡。理论上证明，t 以对数的导数下降时，效果最好。在本实验中所选取的衰减函数是：

$$t_{k+1} = \frac{t_k}{1 + \dfrac{t_k \ln(1+\delta)}{\sigma_k}} \tag{16.8}$$

其中，δ 是一个接近 0 的正的小量，σ_k 是第 k 个 Mapkob 链的目标函数值的方差。

这种衰减函数的特点是，开始时 t 下降很缓慢，越往后 t 下降越快。这对于求得全局最小值是有利的，因为开始时，t 下降很缓慢，有利于算法尝试不同收敛域内的解；而越往后，t 下降越快，使得算法能以较快的速度收敛。

3) 控制参数 t 的终值(停止准则)

一般在接近 0 点时，连续几次降温后，如能量没有明显变化，则终止算法。在本实验中选取的停止准则是：出现连续 s 个 Mapkob 链的结果不变的情况。应用中要注意的是，当问题的规模增大时，应及时调节 s 的值，否则会导致求解时间过长。s 一般小于 2。

4)Mapkob 链的长度 $\overline{L}$

根据 Aarts 等人定义的准平衡条件，当在 t_k 之上达到准平衡，则在 $t_{k=1}$ 上的准平衡只需要进行少量变换就可迅速逼近。如何定义“少量变换”？Aarts 等人指定：该“少量”是指算法能以充分大的概率至少访问给定解的大部分邻域所需进行的变换次数。并且他们证明，对基数为 $|S|$ 的集合进行 N 次重复取样，S 中不同元素被选中的期望概率在 N 和 $|S|$ 为大值时，可近似为：

$$1-e^{-\frac{N}{|S|}}$$

根据以上原理，可将 $\overline{L}$ 取为问题的邻域规模 Θ。但在编程实现中，由于采用是 2 变换和 3 变换随机轮流产生新解，$\overline{L}$ 值大，理论上有利于问题最优解的求得，但是实际上将花费很长的时间。同时，随着问题规模的增大，Θ 增大很快，这对于求解是不利的。因此，在实际的求解过程中，所采取的是将 $\overline{L}$ 取为 2 变换的邻域规模。

由以上步骤可得模拟退火算法的运算过程如下。

①取初始值：i_0，t_0；

②产生新解 j：

③计算目标函数差；

④判断是否接受新解，若不接受，则转回第②步；

⑤接受新解，判断链长是否达到 L_k，若没有，则返回第②步；

⑥减小控制参数 t，重复步骤②~⑤；

⑦判断是否满足停止规则，若不满足则返回第②步。

16.3.3 求解 TSP 问题的程序实现

我们已在 MATLAB 环境下的 SIMULINK 中，将上述利用模拟退火算法求解 TSP 问题用图解和程序实现，并实际求解出来。通过运行程序，可以进行任意节点之间的 TSP 的求解及其图解。下面首先介绍一下程序的界面。应用程序的名字为 SAA4tsp.m，它是用 MATLAB 语言编制的，该程序的界面如图 16.1 所示。

图 16.1 中标签的意义如下：

(1)动态显示求解过程中所产生的路径，控制参数改变一次，该图更新一次。

(2)最终的结果，包括求得的最短路程、求解过程中总共接受的变换次数。

(3)城市数目。

(4)获得城市坐标和城市间距离的方式。城市坐标用一个 $n\times2$ 的矩阵表示，并且在程序中将其命名为 xy，城市间的距离矩阵用 $n\times n$ 的矩阵表示，变量名为 Dis。该下拉菜单有三个选项：

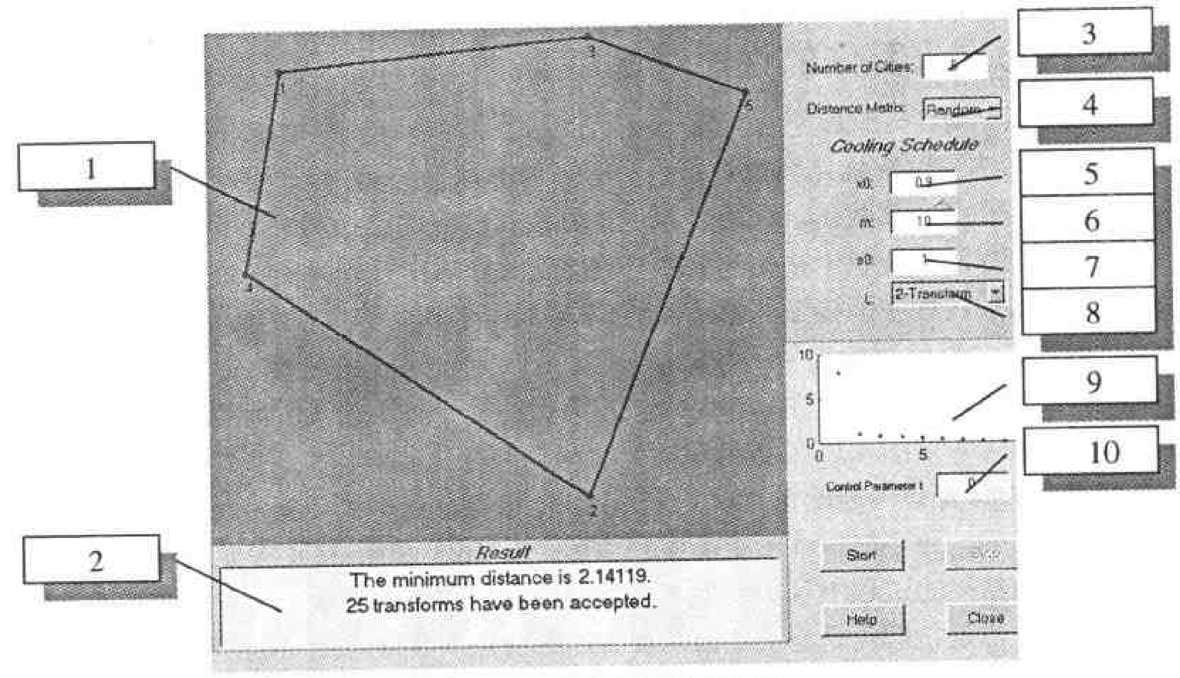


图 16.1　求解 TSP 的界面图

①"Random"，程序将随机自动产生指定数目的城市坐标，并计算出相应的距离矩阵；

②"xydata.mat"，程序将从该文件中提取城市坐标变量（*xy*）。若该文件中含有距离矩阵变量（Dis），则程序直接调用该变量的值；否则，程序将根据 *xy* 计算出 Dis。程序在得出最终结果或被终止时将把变量 *xy* 和 Dis 保存在该文件中。

③"Others…"，指定程序从其他文件中提取有关城市坐标和距离矩阵的信息，该文件中必须有 *xy* 变量。

(5)初始接受概率 χ（应当尽量接近 1）。

(6)为计算控制参数初始值而需要的尝试变换次数（m）。

(7)停止准则：连续 s_0 个 Mapkob 链的结果不变则整个计算停止。

(8)Mapkob 链的长度 $\overline{L}$ 采用 2 变换的规模还是 3 变换的规模。如前所述，当 $\overline{L}$ 取问题的邻域的规模时，我们能尽可能地取到该邻城内的所有值。但是，该程序采用 2 变换和 3 变换交替产生新解，因此 $\overline{L}$ 的取值就成为一个问题。实际应用时将其简单化为或者取 2 变换的邻域规模，或者取 3 变换的邻域规模。当 $\overline{L}$ 取 3 变换的邻域规模时，计算时间比 $\overline{L}$ 取 2 变换的邻域规模时所需时间长，但是解的质量高。

(9)控制参数 t 的变化情况。

(10)当前的控制参数值。图 16.1 中所显示的当前的控制参数值为 0，整个图形所表示的路径是在上一个控制参数控制之下获得的最终解。

程序在退火完成时或被用户终止时，将自动产生两个".mat"文件：xydata.mat，result.mat。"xydata.mat"中保存两个变量：Dis（距离矩阵）和 *xy*（城市坐标）。"result.mat"中保存 3 个变量：fCell、tArray 和 ro。其中，"fCell"是一个 cell 变量，它保存了退火过程中所有产生并被接受了的解所对应的目标函数值；tArray 是一个向量，它保存了控制参数在退火过程中的所有取值；"ro"是程序结束时得到的路径。

16.3.4 模拟退火算法的性能对比

模拟退火算法在解决优化问题上是一种有效的方法。与第 4 章中利用连续 Hopfield 网络（CHNN）求解 TSP 相比，采用模拟退火算法有以下优点：

①模拟退火算法不依赖于初始解。尽管模拟退火算法在开始时需要产生一个初始解，但是由于它在求解过程中能接受坏解，因此它不会局限于初始解所在的收敛域内。而 CHNN 则依赖于初始解，当问题的能量函数的“地形”凹凸不平时，CHNN 得到的结果只能位于初始解所在的收敛域内。下面以 10 个城市为例进行一些比较（所用的城市坐标以及距离矩阵等数据与第 4 章图 4.15 的数据相同）。

图 16.2 的结果是该问题的最优解。采用 SAA 很容易就得到该解，相比之下，CHNN 则要花费很长时间，还不能保证一定能取到该解。图 16.3 和图 16.4 是从两种不同初始值得到相同最优的结果的过程中目标函数（或能量函数）的变化曲线。尽管这两幅图不能全面地说明问题，但是它们还是能反映出两种方法的特点。其一，图 16.3 的曲线比图 16.4 的曲线波动频繁；其二，图 16.3 的曲线的波动范围比图 16.4 的大。这两点都说明了模拟退火算法的求解比 CHNN 有更大的选择性，对于初始值则有更小的依赖性。

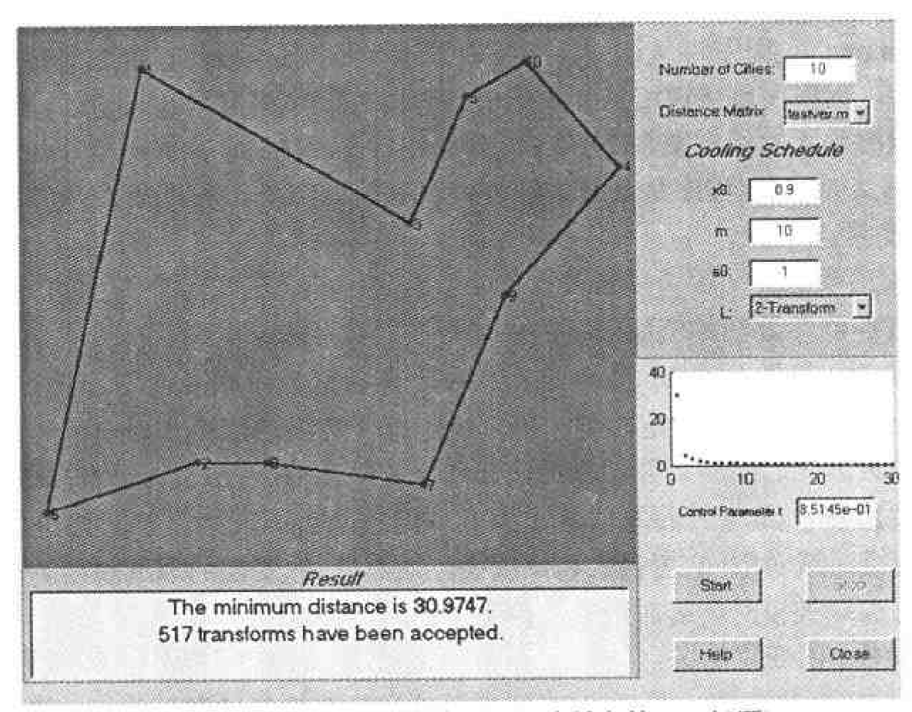


图 16.2 SAA 得出的结果（10 个城市的 TSP 问题）

②模拟退火算法得到的解虽然不一定是最优解，但一定是全局的次优解。原因在于，它在计算过程中，坏解始终都有可能被接受，这使得算法能跳出局部极值的陷阱。CHNN 则不能跳出局部极值的陷阱。也就是说，SAA 比 CHNN 的解的质量高，更有可能得到全局最优值。

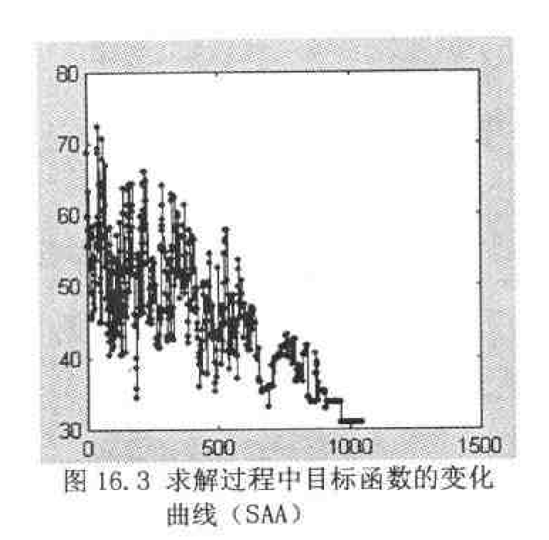


图 16.3 求解过程中目标函数的变化曲线（SAA）

图 16.4 求解过程中目标函数的变化曲线（CHNN）

③模拟退火算法是在问题的解空间内不断产生新解、决定取舍、不断用新解取代旧解从而最终得到问题的一个（次）优解。即 SAA 所产生的解至少都是问题的可行解。CHNN 是在连续的解空间内搜索，这其中涉及到的解不一定是原问题的可行解。因而，CHNN 的搜索空间比 SAA 的大。CHNN 所定义的目标函数既包含了对总距离的限制，又包含着对解的形式的限制，这两者之间必然要有权衡。只有在时间无限长的情况下，CHNN 定义的目标函数才能取到最小值，此时 CHNN 才能得到距离最小的可行解。事实上，我们只能得到次优解，故在所得解中有可能存在非可行解，也就是说，CHNN 要从解中筛选出问题的可行解。从这个意义上说，SAA 比 CHNN 的效率更高。

④CHNN 是一种并行的算法。在不考虑解的质量，只考虑得到一个解的速度时，当城市数目小于 10 时，CHNN 总的来说比 SAA 快，这是很容易理解的。其实，这两种算法正反映了 TSP 这类问题的特点——不存在多项式时间内的求解算法。CHNN 以牺牲空间为代价，换取求解时间的减少；SAA 占用的存储空间较小，但是在求解时间上有所牺牲。在时间和空间之间做出权衡，是求解 TSP 这类问题时必须面对的问题。

综合以上分析，可以看出，SAA 在求解 TSP 这类问题时比 CHNN 更为有效。从实验中，我们可以很容易地看到，SAA 得到的解的质量更高，时间也不一定比 CHNN 所需要的时间长，这是因为 CHNN 要对解进行筛选，并且 CHNN 构造权矩阵的时间也随城市数目的增大而增大。用 CHNN 求解 TSP 问题时，当城市数目大于 30 时，基本上很难得到一个解（时间过长、存储容量过大）。但使用 SAA 可以很轻松地求得 30 个城市的 TSP 问题。图 16.5 就是 35 个城市的例子。

用 MATLAB 编制的 CHNN 的程序求解有名的 144 个城市的 TSP（即 CHN144 实例）几乎是不可能的，但是用 SAA 则可以求出解，不过要花费数小时的时间。图 16.6 给出了利用本章所述方法，采用如图 16.1 所编制的程序与界面，求出 144 个城市的 TSP 的一种解的图形。图 16.6 中显示的结果是算法终止时的解，在求解过程中，得到的最小值是：31167.7。由于以上所有的数据是用 MATLAB 编制的程序来执行的，而 MATLAB 虽然是便于使用的语言，但不是一种高效语言，因此，若换用其他语言会大大提高求解的速度。

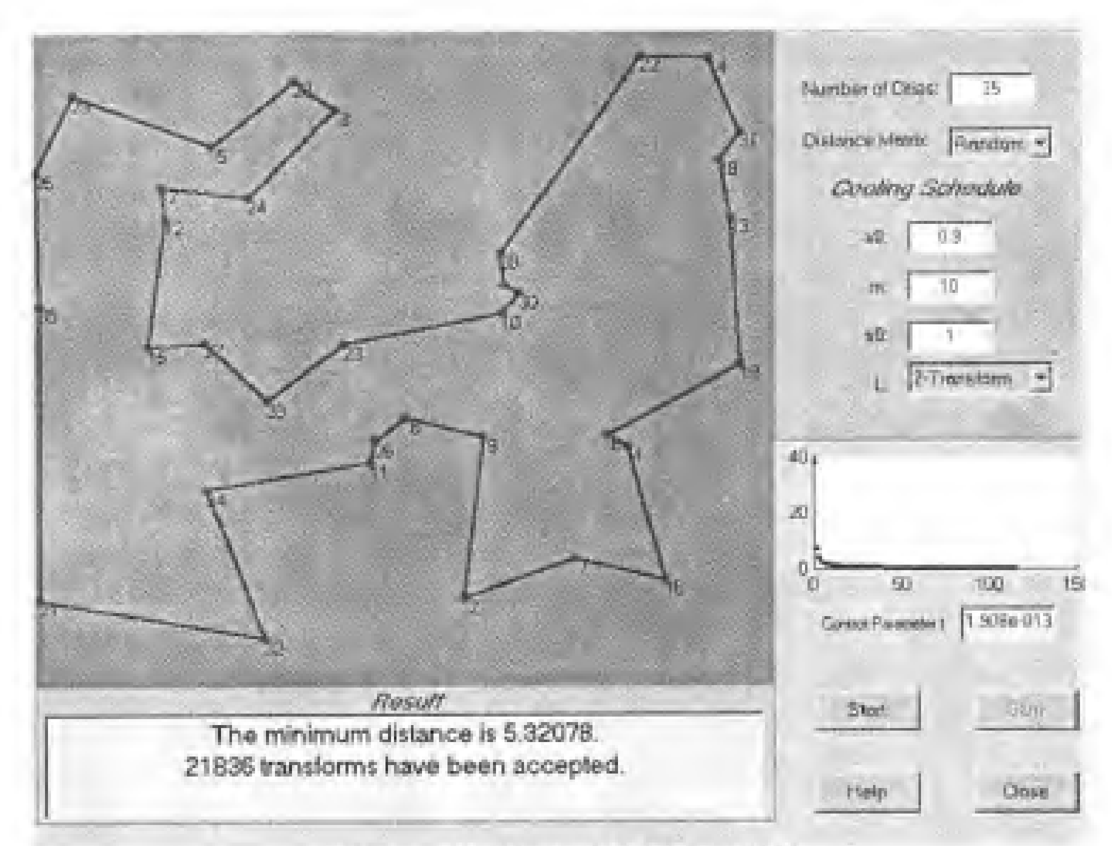


图 16.5 SAA 求解 35 个城市的 TSP 结果

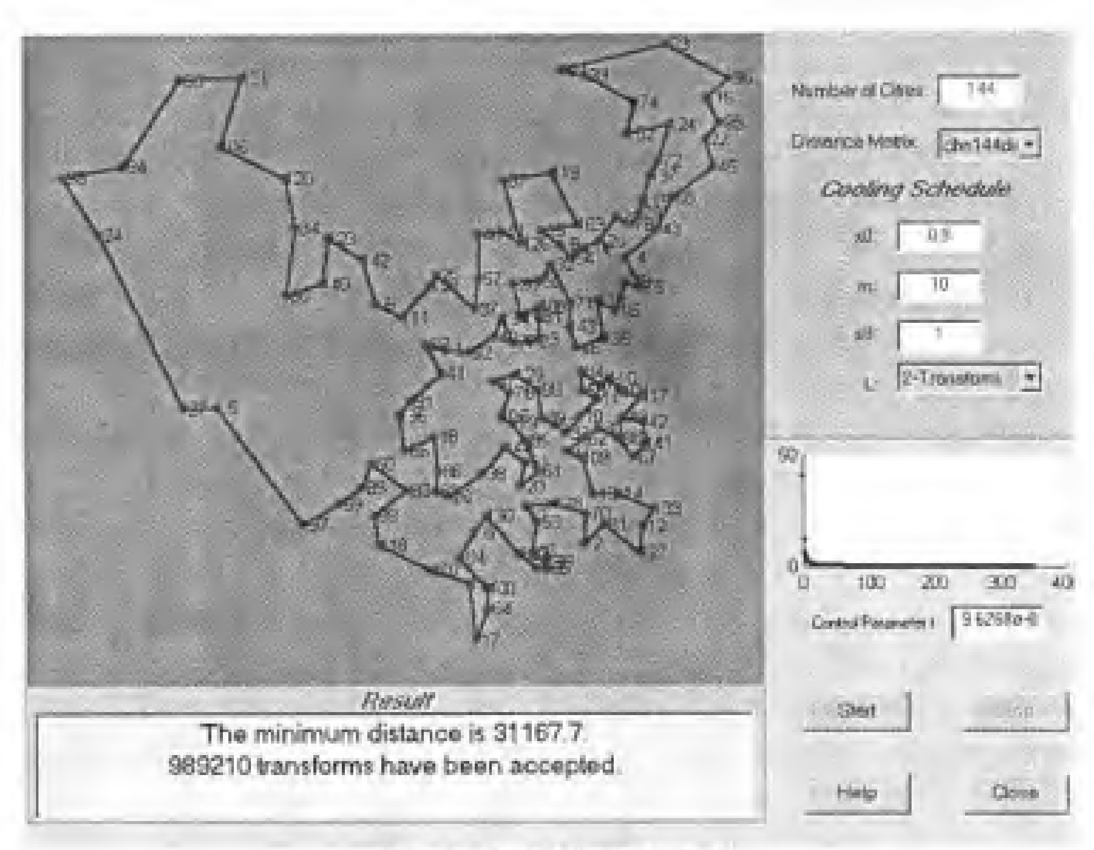


图 16.6 SAA 求解 CHN144 实例

16.4 模拟退火算法的改进

本章讨论的是经典的模拟退火算法，它还有可以改进的地方，尤其是如何确定冷却进度表的参数方面，目前主要有以下几个改进方法。

1）加温退火法

对任意给定的初始解 i_0，先令 $t_0 = 0$，然后进行若干次试验，当且仅当目标函数值增大时接受其转移，同时令 t_0 按某个增量函数 $h(t)$增加，当试验结束时，以所获得的 t_0 值作为控制参数 t 的初始值，并以此时的解 i_0^* 作为初始解开始退火。所需要解决的问题是：t_0 增大到何时为止？

2）有记忆的 SAA

将退火过程中产生的解都记录下来，再与最终解比较，取其最小者。

3）快速模拟退火算法（FSA）

FSA 是针对产生概率服从正态分布的 SAA 提出的，它用 D 维的 Cauchy 分布函数来产生新解，Cauchy 分布与正态分布类似，采用这种分布函数来产生新解的好处是：离旧解越近的地方的解产生的概率越大，这就使得算法在概率上更倾向于局部搜索；同时，采用 Cauchy 分布，离旧解较远的解的产生概率比采用正态分布时的产生概率大，因此，FSA 能够产生较大的“跳跃”。从而，FSA 能缩短求解的时间。本章采用的产生概率是邻域内的均匀分布，它使得算法在概率上是一种邻域内的全局搜索法，并且实现方便。

另外，还有一些带返回搜索的 SA、多次寻优、回火退火法，目前比较新的关于 SAA 的变异方法有：自适应模拟退火（ASA）、Cooperative Simulated Annealing (COSA)等。

参考文献

[1] E. H. L. Aartts, J. H. M. Korst. *Simulated Annealing and Oltzmann Machines*, John Wiley and Sons, 1989

[2] Amontons, G, (1699). *On the Resistance Originating in Machines*, Proc. Of the French Royal Academy of Sciences, pp. 206-222

[3] Armstrong-helouvry B., P. Dupont, C. Canudas Wit. *A Survey of Models, Analysis Tools and Compensation Methods for the Control of Machines with Friction (Survey Paper)*, Automation, 1994, 30(7), 1083-1138

[4] Karl Johan Astrom. *Toward Intelligent Control*, IEEE Control System Magazine, April 1989 pp. 60-64

[5] J. Bottcher, H. R. Traenkler. *Trends in Intelligent/Instrumentation*, IFAC Low Cost Automation 1989, Milan, Italy, 1989, pp. 241-248

[6] Brown, M., and Harris, C. J. *Neurofuzzy Adaptive Modeling and Control*, 1997, Hemel Hempstead, V. K.: Prentice Hall

[7] Cong, Shuang, Strategie Evolute di Controllo Digitale per la Movimentazione Automatica, Tesi di Dottorato di Ricerca, Universita` di Roma, 1995

[8] Cong S. , Li Guodong. *The Decrease of Fuzzy Label Number Using Self-Organization Competition Network*, International ICSC/IFAC Symposium on Neural Computation NC'98, Sep. 23-25, 1998, Vienna, Austria

[9] Da Vinci, L. (1519), *The Notebooks*, Dover, NY

[10] De Carli A., Cong S. and Matacchioni D. *Dynamic Friction Compensation in Servodrives, Proceedings of the 3rd IEEE Conference on Control Applications*, Glasgow, August 1994, 193-198

[11] De Carli, A. and Cong, Shuang. *Strategies for Dynamic Friction Compensation*, European Control Conference 95, Sept. 5-8, 1995, Rome, pp 2730-2735

[12] Alessandro De Carli and Shuang Cong. "*Intelligent Neural Network Controller for a Position Control System*", Proceedings of the IFAC-Workshop "Motion Control", Munich, German, Oct. 1995, 189-196

[13] Peter J. Deasley. *Control: The Integrating Factor in Mechatronics*, Mechatronic Systems Engineering 1, pp. 11-17, 1990

[14] James A.Freeman and David M.Skapura. *Neural network: Algorithms, Applications and Programming Techniques*. Addison-Wesley Publishing Company. 1992

[15] Hao Ying, Yongsheng Ding,ShaoKuan Li and Shihuang Shao. "*Comparison of Necessary Condition for Typical Takagi-Sugeno and Mamdani Fuzzy Systems as Universal Approxmators*", IEEE Trans on System,Man,and Cybernetics,Vol.29,No.5,pp.508-514,1999

[16] Harris C. J. , Wu Z.Q. and Feng M. *Aspects of the Theory and Application of Inteligent Modelling*, Control and Estimation, Proceedings of the 2nd Asian Control Conference, Seoul , 1997, III-1-10

[17] John J.Hopfield and David W.Tank. *Neural Computation of Decisions in Optimization Problems*. Biological Cybernetics, 52: 141-152, 1985

[18] John J.Hopfield and David W.Tank. *Coputing with Neural Circuits: A Model*. Science, 233: 625-633, August 1986

[19] Howard Demuth and Mark Beale. *Neural Network Toolbox User's Guide for Use with MATLAB(Version 3.0)*,The MathWorks, Inc. 1998

[20] K.J. Hunt, D.Sbarbaro, R.Zbikowski and P.J.Gauthrop. *Neural Networks for Control System – A Survey*, Automatica, Vol. 28, No. 6, pp.1083-1112, 1992

[21] Isermann, R. *et al. Adaptive Control Systems*, Prentice Hall International (UK) Ltd., 1992

[22] Rolf Isermann. *Control and Design Aspects for Mechatronic System*, ECC93, pp. 2234-2239

[23] Jan Jantzen. *Tuning-Rules for Fuzzy Controllers*, IEEE International Workshop on Intellignet Motion Control Bogazici University, Istanbul, August 1990, 83-86

[24] Jyh-Shing Roger Jang. "*ANFIS:Adaptive-Network-Based Fuzzy Inference System*", IEEE Trans on System,Man,and Cybernetics,Vol.23,No.3,pp.665-685,1993

[25] Lennart Ljung. *System Identification Toolbox*, User's Guide, Natick, Mass. USA, The MathWorks, Inc., 1993

[26] T. S. Low, T. H. Lee, K. S. Lock and K. J. Jseng. *DSP-Based Instantaneous Torque Control in Perment Magnet Brushless D. C. Drives*, Mechtronics, Vol. 1, No. 2, pp. 203-229, 1991

[27] Morin. A.J. (1833). *New Friction Experiments Carried out at Metz in 1831-1833*, Proc. Of the French Royal Academy of Sciences, 4, 1-128

[28] Kumpati S. Narendra and Kannan Parthasarathy. "*Identification and Control of Dynamical Systems Using Neural Networks*",IEEE Trans on Neural Networks,Vol.1,No.1,pp.4-26,1990

[29] J.-S. Roger Jang and C.-T. Sun. *Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference System*, IEEE Trans. on Neural Networks, 4(1): 156-159, Jan, 1993

[30] Jyh-Shing Roger Jang and Chuen-Tsai Sun. "*Neuro-Fuzzy Modeling and Control*", Proceedings of the IEEE,Vol.83,No.3,1995

[31] Katsuhiko O. *Modern Control Engineering*, Prentice-Hall International, Inc., Second edition, 1990

[32] Reynolds. O. (1886). *On the Theory of Lubrication and its Application to Mr. Beauchamp Tower's Experiments, Including an Experimental Determination of the Viscosity of Olive Oil*, Phil. Trans, Royal Soc., 177, 343-349

[33] Sangbong Park and Cheol Hoon Park. *Adaptive System Identification Using Multilayer Neural Network and Gaussian Potential Function Networks*, IEEE International Conf. On Neural Networks, Vol.4, pp2261-226

[34] Sarle W. S. *ed. Neural Network FAQ*, part 2 of 7: Learning, periodic posting to the usenet newgroup comp.ai.neural-nets(1999)
[35] M. Sugeno. *Industrial Applications Of fuzzy Control*, Elsevier Science Pub Co., 1985
[36] David W.Tank and John J.Hopfield. *Collective Computation in Neuronlike Circuits*. Scientific American, 257(6): 104-114, December 1987
[37] P. D. Wasserman. *Advanced Methods in Neural Computing*, New York, Van Norstrand Reinhold 1993
[38] B. Widrow, M.Bilello. *Nonlinear Adaptive Signal Processing for Inverse Control*, World Conference on Neural Network(WCNN'94), 1994
[39] 方崇智，萧德云. 过程辨识. 北京:清华大学出版社，1998. 2
[40] 孙德敏. 工程最优化方法及应用.合肥：中国科学技术大学出版社，1997
[41] 孙增圻，张再兴，邓志东. 智能控制理论与技术. 北京:清华大学出版社.南宁：广西科学技术出版社，1997.4
[42] 窦振中. 模糊逻辑控制技术及其应用. 北京:北京航空航天大学出版社，1995
[43] 康立山，谢云，尤矢勇，罗祖华. 非数值并行算法——模拟退火算法. 北京:科学出版社，1998.7
[44] 赵振宇，徐用懋. 模糊理论和神经网络的基础与应用. 北京:清华大学出版社，1996.6
[45] 王省富. 样条函数及应用. 1993, 西安：西北工业大学出版社
[46] 杨开宇，陆阂宁. 模糊控制算法的研究. 东南大学学报. 1995, 25(5a): 254-258
[47] 应行仁，曾南. 采用 BP 神经网络记忆模糊规则的控制. 自动化学报，1991,17(1):63-67
[48] 虞和济，陈长征，张省，周建男. 基于神经网络的智能诊断. 北京:冶金工业出版社，2000.5
[49] 张乃饶. 用遗传算法优化模糊器的隶属函数.电气自动化，1996,1: 4-6
[50] 丛 爽. 一种直流伺服驱动器的模糊逻辑控制.控制理论与应用，第 13 卷，1996 年 10 月，176—177
[51] 丛 爽. 运动控制中模糊逻辑控制的应用.电气自动化，1997, 6：16-18
[52] 丛 爽. 面向 MATLAB 工具箱的神经网络理论与应用.合肥：中国科学技术大学出版社，1998.10
[53] 丛 爽，De Carli. A. 两种补偿动态摩擦力的先进控制策略.自动化学报，1998，24(2): 236-240
[54] 丛 爽，钱镇. 采用神经网络和遗传算法优化模糊逻辑控制.中国科学技术大学学报，1998，28：1-5
[55] 丛 爽. 模糊神经网络和遗传算法相结合的控制策略.自动化理论、技术与应用（第五卷），中南工业大学出版社，133-137,1998.7
[56] 丛 爽. 运动控制中先进控制策略的研究综述.微特电机，1998，26(1)：2-8
[57] 丛 爽. 位置跟踪系统中三种控制器的设计与性能比较.微特电机，1998，26(6)：21-23
[58] 丛 爽. 机电系统中模糊与模糊神经元网络控制策略的研究.中国电机工程学报，1999，19(7)：30-32，37

[59] 从 爽，李国栋. 自适应 B 样条模糊神经网络控制器的设计.计算机工程与应用，1999.9：66-68

[60] 从 爽. 机电系统中模糊与模糊神经元网络控制策略的研究.中国电气工程学报，1999，19(7)：30-32，37

[61] 从 爽，赵何. 反向传播网络的不足与改进.自动化博览，1999 年，92(1)：25-26，47

[62] 从 爽. 采用遗传算法提高神经网络模型辨识的精度. 电气自动化，1999 ，21(1): 22-23

[63] 从 爽，李国栋. 自适应 B 样条模糊神经网络控制器的设计.计算机工程与应用，1999，35（9）66-68

[64] 从 爽. 感知器网络的解析、局限性与拓展.自动化博览，2000（3），34-36

[65] 从 爽，李国栋. 并联双模糊控制器在摩擦力补偿中的应用，微特电机，2000,28（3），3-5

[66] 从 爽. 典型人工神经网络的结构、功能及其在智能系统中的应用.信息与控制，2001.2

[67] 高雪鹏，从 爽. BP 网络改进算法的性能对比研究. 控制与决策，2001，16（2），167-171